

NASA-CR-199181

SPECTRAL METHODS IN THE SIMULATION
OF SYSTEMS WITH THERMAL
AND COMPOSITIONAL GRADIENTS

FINAL REPORT ON NASA CONTRACT NAS8-35838

(NASA-CR-199181) SPECTRAL METHODS
IN THE SIMULATION OF SYSTEMS WITH
THERMAL AND COMPOSITIONAL GRADIENTS
Final Report (Hart (Fred C.)
Associates) 113 p

N96-70517

Unclassified

29/64 0064562

PROJECT SUMMARY

NAS8-35838

This report describes and summarizes the development of two general classes of spectral numerical codes for the solution of convection-diffusion problems in several space dimensions. In addition, results obtained with these codes are surveyed and the requirements for extension of these numerical methods to the solution of three-dimensional flow problems are discussed.

no. Spectral methods have clear advantages over more conventional methods provided that certain conditions are met. Efficient direct solutions of spectral equations are generally possible only for relatively simple geometries if implicit methods are used; or for more general geometries if explicit time stepping methods are used. In other cases, it has been possible to devise iterative methods to solve the resulting spectral equations. The key idea is to approximate the full spectral operator with a sparse finite difference operator that may be solved in little more work than is necessary to solve the sparse finite difference equations.

Another key feature of spectral methods relates to their patching properties when solutions in two or more neighboring domains are 'patched' together. If finite difference methods are used in such patched domains, it is well known that it is necessary to utilize overlapping grids to avoid numerical instabilities and accuracies that may ruin the fidelity of a numerical solution. The origin of this behavior of finite difference schemes is related to the dispersive character of waves propagating on distinct finite difference grids. In contrast, spectral solutions may be directly patched together without either stability or accuracy problems. It is not necessary to have overlapping domains of validity for the spectral solutions. Of course, it is necessary to choose appropriate spectral representations in each of the neighboring domains. This report presents, specifically, a program to solve the Boussinesq equations for a doubly diffusive fluid layer. Then three programs to solve diffusion equations of varying degrees of nonlinearity and complexity are described. Included is a code for the solution of Bridgeman-type directional solidification problems by a Chebyshev spectral method. Extensions of this method to more general three-dimensional geometries will permit the solution of otherwise intractable problems in crystal growth dynamics. The resulting codes are quite novel in that they lead to extremely accurate results with minimal spatial resolution.

TABLE OF CONTENTS

1. INTRODUCTION	1
2. NUMERICAL SIMULATION OF DOUBLE DIFFUSIVE CONVECTION	7
3. MULTI-DOMAIN DIFFUSION PROBLEMS	66
4. REFERENCES	101

SPECTRAL METHODS IN THE SIMULATION OF
SYSTEMS WITH THERMAL AND COMPOSITIONAL GRADIENTS

- FINAL REPORT ON NASA CONTRACT NAS8-35838 -

1. INTRODUCTION

In this Report, we summarize and describe the development of two general classes of spectral numerical codes for the solution of convection-diffusion problems in several space dimensions. In addition, we survey results obtained with these codes and discuss the requirements for extension of these numerical methods to the solution of three-dimensional flow problems.

Spectral methods involve the solution of partial differential equations by representing the solution in terms of series of smooth functions. In contrast to finite-element methods in which the solution is expressed in terms of piecewise-smooth functions (which may have derivative discontinuities at the edges of the small finite elements), spectral methods are closer to what may be called global element methods, in which the basis functions are smooth over relatively large regions of space.

The simplest example of a spectral method is given by a 'pseudospectral Fourier' method. This method involves basically two steps to obtain a numerical approximation $u_N(x)$ to the

solution to a differential equation. First, the solution $u_N(x)$ is assumed to be represented in the form of a finite series of complex exponential functions:

$$u_N(x) = \sum_{k=-N/2}^{N/2-1} a_k e^{ikx} \quad (1.1)$$

The second step is to obtain equations for the expansion coefficients a_k from the differential approximation whose solution is approximated by u_N .

Using the discrete Fourier transform, it is easy to see that knowledge of $u_N(x)$ at the N discrete points $x_j = 2\pi j/N$, $j = 0, 1, \dots, N-1$ is equivalent to knowledge of the N expansion coefficients a_k , $k = -N/2, -N/2+1, \dots, N/2-1$. Indeed,

$$a_k = \frac{1}{N} \sum_{j=0}^{N-1} u_N(x_j) e^{-ikx_j} \quad (1.2)$$

[The asymmetry regarding the modes with $k = \pm N/2$ is easily explained if it is noted that reality of $u_N(x_j)$ implies that a_k satisfies the conjugate-symmetry condition

$$a_{-k} = a_k^* \quad (1.3)$$

where $*$ denotes complex conjugate. Thus, $a_{N/2} = a_{-N/2}^*$ is real, since $a_{k+N} = a_k$.]

In order to obtain the equations for the expansion coefficients a_k [or, equivalently, for the collocation grid point values $u_N(x_j)$] the crucial step is to obtain values for the derivatives $d^p u_N(x)/dx^p$ at the collocation points x_j in terms of the values $u_N(x_j)$. It is at this step that spectral methods differ from the more elementary finite difference methods. In finite difference

schemes, the expression for derivative in terms of the collocation grid point values is only approximate; in spectral methods, the expression for derivatives in terms of the collocation grid is exact.

In the pseudospectral Fourier method, the expression for derivatives of u_N at the collocation grid points x_j is simply obtained by differentiating (1.1):

$$\frac{d^p u_N(x_j)}{dx^p} = \sum_{k=-N/2}^{N/2-1} (ik)^p a_k e^{ikx_j} \quad (1.4)$$

Since the expansion coefficients a_k are known in terms of the collocation grid point values of u_N , it follows that all derivatives of u_N are similarly known in terms of $u_N(x_j)$. Once these expressions for derivatives are known, they may be directly applied to obtain dynamical equations for either a_k or $u_N(x_j)$ from the basic governing differential equation. For example, consider the linear differential equation

$$u'' + a(x)u' + b(x)u = c(x) \quad (1.5)$$

with 2π -periodic coefficients and boundary conditions. The pseudospectral Fourier equations for the approximation u_N based on the points x_j are simply

$$\frac{d^2 u_N(x_j)}{dx^2} + a(x_j) \frac{du_N(x_j)}{dx} + b(x_j)u_N(x_j) = c(x_j) \quad (1.6)$$

Each of the terms appearing in (1.6) is easily evaluated by fast discrete transform techniques based on the fast Fourier transform.

Spectral methods constructed in this way have clear advantages over more conventional methods provided that the discrete spectral equations can be solved efficiently and that the solution to the continuous problem is well behaved. Efficient direct solutions of spectral equations are generally possible only for relatively simple geometries if implicit methods are used or for more general geometries if explicit time stepping methods are used. In other cases, it has been possible to devise iterative methods to solve the resulting spectral equations. The key idea is to approximate the full spectral operator with a sparse finite difference operator, following Orszag (1980). The basic idea is that the sparse finite difference operator may be constructed to be within a bounded factor of the more accurate spectral operator on all resolved spatial scales. This fact allows the construction of iterative methods that are guaranteed of convergence in a finite number of iterations -- the result is that the spectral equations may be solved in little more work than is necessary to solve the sparse finite difference equations.

Another key feature of spectral methods relates to their patching properties when solutions in two or more neighboring domains are 'patched' together. If finite difference methods are used in such patched domains, it is well known that it is necessary to utilize overlapping grids to avoid numerical instabilities and accuracies that may ruin the fidelity of a numerical solution. The origin of this behavior of finite difference schemes is related to the dispersive character of waves propagating on distinct finite difference grids. In contrast, spectral solutions

may be directly patched together without either stability or accuracy problems. It is not necessary to have overlapping domains of validity for the spectral solutions. Of course, it is necessary to choose appropriate spectral representations in each of the neighboring domains -- the spectral representations chosen must be compatible with the boundary conditions or continuity conditions that must be applied at the edges of each domain. For example, in the solution of the multi-domain heat equation

$$\frac{\partial u(x,t)}{\partial t} = \frac{\partial}{\partial x} k_1(x,t) \frac{\partial u}{\partial x} \quad (x < 0) \quad (1.7)$$

$$\frac{\partial u(x,t)}{\partial t} = \frac{\partial}{\partial x} k_2(x,t) \frac{\partial u}{\partial x} \quad (x > 0)$$

where k_1, k_2 are distinct heat conductivity coefficients in their respective domains, it is necessary to apply the conditions of continuity of $u_N(x,t)$ and heat flux at the boundary $x=0$ between the neighboring domains:

$$u_N(0-,t) = u_N(0+,t) \quad (1.8)$$

$$k_1(0,t) \frac{\partial u_N(0-,t)}{\partial x} = k_2(0,t) \frac{\partial u_N(0+,t)}{\partial x}$$

In order to impose the conditions (1.8) properly, it is necessary to use an appropriate spectral series representation of $u(x,t)$. For this purpose, complex exponential series are not satisfactory, but polynomial series, like Chebyshev polynomial series or Legendre polynomial series, are excellent.

In the remainder of this Report, we describe computer codes for the solution of multi-dimensional fluid flow and heat conduction problems based on the above-described ideas. The resulting codes are quite novel in that they lead to extremely accurate results with minimal spatial resolution. They have been

applied to a variety of test problems and production models. For typical problems, these spectral codes require between one and three orders of magnitude fewer degrees of freedom to describe the dynamical system than is required by competitive, high accuracy, finite difference models of the same differential equations. Specific results and comparisons are given by Shih & Orszag (1984) and Moro & Orszag (1984).

In Chapter 2, the program DBLDIFF to solve the Boussinesq equations for a doubly diffusive fluid layer is described. Then, in Chapter 3, three programs to solve diffusion oequations of varying degrees of nonlinearity and complexity are described. Included here is a code for the solution of Bridgman-type directional solidification problems by a Chebyshev spectral method. We believe that extensions of this method to more general three-dimensional geometries will permit the solution of otherwise intractable problems in crystal growth dynamics.

2. NUMERICAL SIMULATION OF DOUBLE DIFFUSIVE CONVECTION

The equations of motion of a double diffusive Boussinesq fluid layer are

$$\frac{\partial \vec{v}}{\partial t} = \vec{v} \times \vec{\omega} - \nabla(p + \frac{1}{2}\nu) + (\alpha_1 T + \alpha_2 S) \hat{z} + \nu \nabla^2 \vec{v} \quad (2.1)$$

$$\frac{\partial T}{\partial t} + \vec{v} \cdot \nabla T = \beta_1 w + \kappa_1 \nabla^2 T \quad (2.2)$$

$$\frac{\partial S}{\partial t} + \vec{v} \cdot \nabla S = \beta_2 w + \kappa_2 \nabla^2 S \quad (2.3)$$

$$\nabla \cdot \vec{v} = 0 \quad (2.4)$$

Here $\vec{v} = (u, v, w)$ is the velocity field at $\vec{x} = (x, y, z)$, $\vec{\omega} = \vec{v} \times \vec{v}$ is the vorticity, p is the pressure (assuming that the density is normalized to unity), T is the deviation of the temperature from the conduction profile $-\beta_1 z$ (so that $T - \beta_1 z$ is the actual temperature), S is the deviation of the salinity from the conduction profile (so that $S - \beta_2 z$ is the actual salinity), ν is the kinematic viscosity, κ_1 and κ_2 are the conductivities of heat and salinity, respectively, and α_1 and α_2 are the respective expansion coefficients. The flow is assumed to occur in the region

$$0 \leq x \leq \lambda H, \quad 0 \leq z \leq H \quad (2.5)$$

Periodic boundary conditions are applied in x with period

$$\lambda H: \quad \vec{v}(x + rH, z) = \vec{v}(x, z) \quad (2.6)$$

Free slip (no-stress) impermeable boundary conditions are applied at $z = 0, H$, so

$$\frac{\partial u}{\partial z} = w = 0 \quad (z=0, H) \quad (2.7)$$

while T and S are assumed to satisfy the conducting-boundary conditions

$$T = S = 0 \quad (z=0, H) \quad (2.8)$$

Also, $\beta_1 < 0$ corresponds to a thermally stable layer, while $\beta_2 > 0$ gives an unstably stratified saline layer. In many case of interest, this combination of signs of β_1 and β_2 is realized.

We solve (2.1-7) in terms of a spectral representation of the form

$$\begin{Bmatrix} U \\ W \\ T \\ S \end{Bmatrix}(x, z, t) = \sum_{m=-\frac{1}{2}M}^{\frac{1}{2}M-1} \sum_{p=0}^P \begin{Bmatrix} \hat{U} \\ \hat{W} \\ \hat{T} \\ \hat{S} \end{Bmatrix}(m, p, t) e^{2\pi i mx/\lambda} \begin{Bmatrix} \cos \pi p z \\ \sin \pi p z \\ \sin \pi p z \\ \sin \pi p z \end{Bmatrix} \quad (2.9)$$

In the following fully documented and self-explanatory code for the solution of these equations, the nonlinear terms are evaluated by fast-transform methods with aliasing terms usually removed. Time stepping is done by a leapfrog scheme for the nonlinear terms and an implicit scheme for the viscous terms (which may be either the Crank-Nicolson or backwards Euler method depending on user preference). The pressure term is computed in Fourier representation by local algebraic manipulation of the incompressibility constraint (2.4).

The resulting computer code is extremely efficient.

In two space dimensions, typical computer times are about 0.21 s per time step on a Cray-1 computer or about 3s per time step on a Cyber 175, using spatial resolution of $M = P = 128$ in (2.9). Note that the code presented on the following pages actually solves (2.1-7) in terms of a vorticity-streamfunction

representation of the velocity field. However, the basic ideas used in generating this computer code do generalize to fully three dimensional flows. We believe that it is now, on the basis of our experience in generating this code for doubly diffusive convection, straightforward to generate a fully three-dimensional spectral code that is both very efficient and accurate.

We also note that the code given on the following pages has the attractive features that:

- a. Aliasing interactions may be controlled by the user. The code may run in any configuration between fully pseudospectral and fully spectral may simply changing a programmer accessible switch.
- b. Time stepping is done by the efficient leapfrog scheme on the nonlinear terms.
- c. The code is easily portable between the Cyber 175 series computers and the Cray-1 series computers.
- d. Fully integrated graphical and numerical output programs are presented. The code is stand-alone with standard graphics packages.
- e. The code is self-documenting.

```
*****
DOUBLE DIFFUSIVE OCEAN FORTRAN PROGRAMS
*****
```

```
PROGRAM DBLDIFF
```

```
) (Z)/DT + (U,V).GRAD(Z) = RNU*DELSQ(Z) + ALPHA1*D(TEMP)/DX + ALPHA2*DS/ DX
) (TEMP)/DT + (U,V).GRAD(TEMP) = RKAPA1*DELSQ(TEMP) + BETA1*V
) (SALT)/DT + (U,V).GRAD(SALT) = RKAPA2*DELSQ(SALT) + BETA2*V
```

THIS PROGRAM SIMULATES TWO-DIMENSIONAL FLUID FLOW ON A $(N_2+1) \times N_1$ GRID. IT IS POSSIBLE TO SPECIFY NON-LINEAR DISTRIBUTION OF DATA POINTS IN THE Y-DIRECTION (THE FIRST INDEX CORRESPONDS TO Y DIRECTION, SECOND TO X DIRECTION). N_1 AND N_2 HAVE TO BE POWERS OF TWO GREATER OR EQUAL TO 8

THE PROGRAM USES LARGE CORE MEMORY. IT REQUIRES $8*(N_2+1)*N_1+(N_2-1)*(N_2$ WORDS OF IT. IF NOTEMP = 1 IT IS POSSIBLE TO SAVE $2*(N_2+1)*N_1$ WORDS, IF LMAPP = 1 YOU SAVE ADDITIONAL $(N_2-1)*(N_2+1)$ WORDS.

IF LMAPP = 2, LCM REQUIREMENTS ARE AT LEAST $3 * (N_2-1) * (N_2-1)$ WORDS. MORE MEMORY COULD BE NEEDED IN THIS CASE BECAUSE OF EIGSET AND LCMRQR.

THE FOLLOWING PARAMETERS DEFINE THE SIZE OF THE RUN:

```
PARAMETER (NX=16, NY=16)
PARAMETER (NA1=2*(NY+1), NA2=NX/2, NE=NY-1, NW1=NY+1)
PARAMETER (NW2=128, NEX=256)
PARAMETER (NI=2*NY-1, NEQ=3*NY-2, NCW2=2*NW1+1, NCW3=4*NW1+1)
```

THESE REFER TO THE DIMENSIONING AND EQUIVALENCING INSTRUCTIONS IN THE SOURCE CODE. YOUR COMPILER CANNOT HANDLE PARAMETER STATEMENTS... COMMON BLOCKS:

```
COMMON /FT/ NPTS, NSKIP, MTRN, MSKIP, ISIGN, LOG, IEXSHFT
```

THE DIMENSION OF GSOLV HAS TO BE AT LEAST (N_2-1)

```
COMMON /SOLV/ LVECT, LSKIP, NVECT, NVSKIP, GSOLV(127)
```

THE DIMENSIONS OF GMAPP AND G HAVE TO BE AT LEAST (N_2+1)

```
COMMON /MAPPING/ GMAPP(129), G(129)
COMMON /MFLOW/ U(129)
COMMON /DATA/ ALPHA1, ALPHA2, BETA1, BETA2, BOXX, H, RNU,
RKAPA1, RKAPA2, DT, DTT, T, NSTEPS
COMMON /OPTIONS/ N1, N2, NUMBER, KDATA, LMAPP, NOTEMP, KTA, KTAPE,
1 KSM, KPP, KZP, KTP, KUP, KVP, KPF, KZF, KTF, KUF, KVF,
2 KPRESS, NOALS, NOSALT
COMMON /PI/ PI
COMMON /NMBRS/ N1H, N1Q, N1E, N2M, N2P, N2DP, N2DM, N2PD
COMMON /EXSIZE/ NMAX2
COMMON /LADDR/ LZN, LTN, LPN, LZO, LTO, LPY, LZX, LZY, LEIG, LE, LH, LEV,
1 LSN, LSO
```

ORIGINAL PAGE IS
OF POOR QUALITY

Jul 13 18:46 1984 dbldif Page 2

```

C REAL ARRAYS HAVE TO BE DIMENSIONED EXACTLY AS THE FORMULAS SAY:
C     A      2*(N2+1) X N1/2
C     E      (N2-1)   X (N2-1)
C     WORK   N2+1    X MAXØ(6, 2*MINØ(64, N1/2)) (AT LEAST)
C     EVAL   N2-1
C     IR     N2-1
C     IC     N2-1
C
C     REAL A(NA1,NA2), E(NE,NE)
C     REAL WORK(NW1,NW2)
C     REAL EVAL(NE), IR(NE), IC(NE)
C
C COMPLEX ARRAYS HAVE TO BE DIMENSIONED EXACTLY AS THE FORMULAS SAY:
C     CA      (N2+1) X N1/2
C     CWORK   N2+1
C     CWORK2  N2+1
C     CWORK3  N2+1
C     EX      MAXØ(2*N2, N1)
C
C EX ARRAY MUST NOT BE THE FIRST ARRAY IN THE MEMORY.
C
C     COMPLEX CA(NW1,NA2), CWORK(NW1), CWORK2(NW1)
C     COMPLEX CWORK3(NW1), EX(NEX), CF
C
C EQUIVALENCE STATEMENTS ARE SAVING MEMORY SPACE. THE FOLLOWING PATTERN
C HAS TO BE FOLLOWED EXACTLY:
C
C     A-----+-----+-----+-----+-----+-----+-----+-----+
C     CA-----+-----+-----+-----+-----+-----+-----+-----+
C     EVAL---+IR---+IC---+E---+-----+-----+-----+-----+
C
C
C     WORK---+-----+-----+-----+-----+
C     CWORK---+-----+CWORK2---+-----+CWORK3---+
C
C     EQUIVALENCE (A,CA,EVAL), (A(NY),IR), (A(NI),IC), (A(NEQ),E)
C     EQUIVALENCE (WORK,CWORK), (WORK(NCW2),CWORK2), (WORK(NCW3),CWORK3)
C
C PI IS A USEFULL CONSTANT
C
C     DATA PI /3.141592653589793/
C
C NOW WE CAN START WITH THE PROGRAM...
C SUBROUTINE START TELLS OCEAN WHAT TO DO:
C
C     CALL START (NX,NY)
C
C IF LMAPP = 1 , H (THE SIZE OF THE BOX IN Y DIRECTION) SHOULD BE
C SPECIFIED IN THE MAIN PROGRAM.
C
C     H = PI
C
C DEFINING CONSTANTS THAT WILL BE NEEDED:
C
C     N1H = N1/2
C     N1Q = N1/4
C     N1E = N1/8
C     N2M = N2-1
C     N2P = N2+1
C     N2DP = 2*N2+1
C     N2DM = 2*N2-1
C     N2PD = 2*N2P

```

NW = N2PD*N1H
 Jul 13 18:46 1984 db1aif Page 3

```

NWH = NW/2
NWQ = NW/4
NWEIG = N2M
IF (LMAPP .NE. 1) NWEIG = (N2M+3)*N2M
NMAX2 = MAX0(2*N2,N1)
K1Q = N1E+1
K2Q = K1Q+N1E
K3Q = K2Q+N1E
K1QM = K1Q-1
K2QM = K2Q-1
K3QM = K3Q-1
C
C LCM ALLOCATION AND COMPUTATION OF ADDRESSES:
C
  LARGE = 10*NW
  IF (NOSALT.EQ.1)LARGE=LARGE-2*NW
  IF (NOTEMP .EQ. 1) LARGE = LARGE-2*NW
  IF (LMAPP .NE. 2) GO TO 66
C
  IF (KPRESS .GT. 1) LARGE = MAX0 (LARGE, 3*N2M*N2M) +NWEIG+NWEIG
  IF (KPRESS .LE. 1) LARGE = MAX0 (LARGE+NWEIG, 3*N2M*N2M)
  GO TO 67
C
  66 LARGE = LARGE + NWEIG
C
  67 LZN = LCMREQ (LARGE)
  LTN = LZN+NW
  IF (NOTEMP .EQ. 1) LTN = LZN
  LPN = LTN+NW
  LZO = LPN+NW
  LTO = LZO+NW
  IF (NOTEMP .EQ. 1) LTO = LZO
  LPY = LTO+NW
  LSN=LPY+NW
  LSO=LSN+NW
  LZX = LSO+NW
  IF (NOSALT.EQ.1)LZX=LSN
  LZY = LZX+NW
  LPRESS = LZN + LARGE - NWEIG
  LEIG = LPRESS
  IF ((KPRESS .GT. 1) .AND. (LMAPP .NE. 1)) LEIG = LPRESS - NWEIG

```

THESE LCM ADDRESSES ARE USED BY EIGSET AND LCMRQR

```

C
  LE = LZN
  LH = LE + N2M*N2M
  LEV = LH + N2M*N2M
C
  CALL EXSET (EX,NMAX2)
C
  SET ALL U(I) TO Ø. TO AVOID TROUBLE IF NO MEAN FLOW IS DESIRED
C
  DO 11 I = 1,N2P
  11    U(I) = Ø.
C
  SET UP ARRAYS ASSOCIATED WITH THE MAPPING FUNCTION:
C
  IF (LMAPP .NE. 1) GO TO 2
C
  LINEAR MAPPING - EIGENVALUES KNOWN, EIGENFUNCTIONS ARE SIMPLY SINES
  AND COSINES, SO THAT NO MATRICES ARE NEEDED:

```

C F = H/FLOAT(N2)

JUL 13 18:46 1984 dbldif Page 4

```

FZ = PI/H
FZSQ = FZ*FZ
GMAPP(1) = 0.
G(1) = FZ
DO 1 I = 1,N2M
    GMAPP(I+1) = F*FLOAT(I)
    G(I+1) = FZ
1     EVAL(I) = -(FLOAT(I)**2)*FZSQ
    GMAPP(N2P) = H
    G(N2P) = FZ
    CALL OUT (EVAL,LEIG,NWEIG)
    GO TO 4
2     IF (LMAPP .NE. 2) GO TO 3
C NONLINEAR MAPPING - FIND OUT WHAT IS IT AND SET UP THE VISCOS MATRIX:
C
CALL GSET (GMAPP)
CALL EIGSET (CA,G,GMAPP,CWORK,EX,EVAL,IR,IC,E,N2M)
C SET UP THE PRESSURE MATRIX:
C
IF (KPRESS .GT. 1) CALL PRESSET (CA, G, GMAPP, CWORK, EX,
1                               EVAL, IR, IC, E, N2M, LPRESS)
    GO TO 4
3     IF (LMAPP .NE. 3) STOP LMAP
    CALL EIGIN (EVAL,IR,IC,E,G,GMAPP)
4     IF ((LMAPP .EQ. 2) .AND. ((KTAPE .EQ. 1) .OR. (KTAPE .EQ. 3)))
1     CALL EIGOUT (EVAL,IR,IC,E,G,GMAPP)
C READ THE DATA:
C
IF ((KDATA .NE. 1) .AND. (KDATA .NE. 2)) GO TO 13
    CALL DATASET (A,N2P,N1,GMAPP)
    GO TO 14
13    IF (KDATA .NE. 3) STOP KDAT
    CALL DATAIN (A,WORK)
14    IF ((KDATA .EQ. 3) .AND. ((KTAPE .EQ. 2) .OR. (KTAPE .EQ. 3)))
1     CALL DATAOUT (A,WORK)
    FX = 2.0*PI/BOXX
    FX2 = 2.*FX

C PREPARE THE INPUT FOR PROCESSING:
C
IF (KDATA .NE. 1) GO TO 40
C THE INPUT WAS STREAMFUNCTION IN PHYSICAL SPACE - SHUFFLE THE DATA AND
C GO TO FOURIER SPACE:
C
CALL IN (A,LPN,NW)
CALL SHUF (A,N2P,N1,WORK)
NPTS = N1
NSKIP = N2P
MTRN = N2P
MSKIP = 1
ISIGN = -1
LOG = LOG2 (NPTS)
IEXSHFT = NMAX2/NPTS
CALL RCS (A,EX)
NPTS = 2*N2
NSKIP = 1

```

MTRN = N1H
 MSKIP = N2P
 ISIGN = -1

Jul 13 18:46 1984 dbldif Page 5

```

LOG = LOG2 (NPTS)
IEXSHFT = NMAX2/NPTS
CALL ACAC (A,EX,WORK)
F = .125/FLOAT(N1*N2)
DO 30 J = 1,N1H
  DO 30 I = 1,N2PD
    A(I,J) = A(I,J)*F
  CALL OUT (A,LPN,NW)
  CALL OUT (A,LZN,NW)
40 IF (KDATA .EQ. 3) GO TO 145
C
C STREAMFUNCTION KNOWN - FIND THE VORTICITY: Z = DEL2 (PSI):
C
  CALL IN (EVAL,LEIG,NWEIG)
  KZN = LZN
  LVECT = N2M
  LSKIP = 2
  NVECT = 2
  NVSKIP = 1
  DO 60 J = 1,N1H
    CALL IN (CWORK,KZN,N2PD)
    IF (LMAPP .NE. 1) CALL SOLVE2 (CWORK(2),E,N2M,IR,IC)
    F = FLOAT((J-1)*(J-1))*FX*FX
    DO 50 I = 2,N2
      CWORK(I) = (EVAL(I-1)-F)*CWORK(I)
      IF (LMAPP .NE. 1) CALL LUMULT2 (CWORK(2),E,N2M,IR,IC)
      CWORK(1) = -F*CWORK(1)
      CWORK(N2P) = .0.
      CALL OUT (CWORK,KZN,N2PD)
50  KZN = KZN+N2PD
60  NPTS = 2*N2
  NSKIP = 1
  MTRN = N1H
  MSKIP = N2P
  ISIGN = 1
  LOG = LOG2(NPTS)
  IEXSHFT = NMAX2/NPTS
  CALL IN (A,LZN,NW)
  CALL ACAC (A,EX,WORK)
  DO 70 J = 1,N1H
    DO 70 I = 1,N2PD
      A(I,J) = .5*A(I,J)
    CALL OUT (A,LZO,NW)
70  IF (NOTEMP .EQ. 1) GO TO 1140
    IF (KDATA .NE. 1) GO TO 1120
C
C SALT IS SUPPLIED IN PHYSICAL SPACE - SHUFFLE THE DATA AND
C GO TO FOURIER REPRESENTATION:
C
  CALL IN (A,LSN,NW)
  CALL SHUF (A,N2P,N1,WORK)
  NPTS = N1
  NSKIP = N2P
  MTRN = N2P
  MSKIP = 1
  ISIGN = -1
  LOG = LOG2 (NPTS)
  IEXSHFT = NMAX2/NPTS
  CALL RCS (A,EX)

```

```

F = 0.5/FLOAT(N1)
DO 1100 J = 1,N1H
    DO 1100 I = 1,N2PD
        A(I,J) = F*A(I,J)
1100

```

Jul 13 18:46 1984 dbldif Page 6

```

CALL OUT (A,LS0,NW)
NPTS = 2*N2
NSKIP = 1
MTRN = N1H
MSKIP = N2P
ISIGN = -1
LOG = LOG2 (NPTS)
IEXSHFT = NMAX2/NPTS
CALL ACAC (A,EX,WORK)
F = 0.25/FLOAT(N2)
DO 1110 J = 1,N1H
    DO 1110 I = 1,N2PD
        A(I,J) = F*A(I,J)
1110
    CALL OUT (A,LSN,NW)
1120 GO TO 1140
    CALL IN (A,LSN,NW)
C
C NEW SALT KNOWN - SET UP S(OLD):
C
NPTS = 2*N2
NSKIP = 1
MTRN = N1H
MSKIP = N2P
ISIGN = 1
LOG = LOG2 (NPTS)
IEXSHFT = NMAX2/NPTS
CALL ACAC (A,EX,WORK)
DO 1130 J = 1,N1H
    DO 1130 I = 1,N2PD
        A(I,J) = A(I,J)*0.5
1130
    CALL OUT (A,LS0,NW)
1140 IF (NOTEMP .EQ. 1) GO TO 140
    IF (KDATA .NE. 1) GO TO 120
C
C TEMPERATURE IS SUPPLIED IN PHYSICAL SPACE - SHUFFLE THE DATA AND
C GO TO FOURIER REPRESENTATION:
C
    CALL IN (A,LTN,NW)
    CALL SHUF (A,N2P,N1,WORK)
    NPTS = N1
    NSKIP = N2P
    MTRN = N2P
    MSKIP = 1
    ISIGN = -1
    LOG = LOG2 (NPTS)
    IEXSHFT = NMAX2/NPTS
    CALL RCS (A,EX)
    F = 0.5/FLOAT(N1)
    DO 100 J = 1,N1H
        DO 100 I = 1,N2PD
            A(I,J) = F*A(I,J)
100
    CALL OUT (A,LTO,NW)
    NPTS = 2*N2
    NSKIP = 1
    MTRN = N1H
    MSKIP = N2P
    ISIGN = -1
    LOG = LOG2 (NPTS)

```

```

IEXSHFT = NMAX2/NPTS
CALL ACAC (A,EX,WORK)
F = .25/FLOAT(N2)
DO 110 J = 1,N1H
    DO 110 I = 1,N2PD

```

Jul 13 18:46 1984 dbldif Page 7

```

110          A(I,J) = F*A(I,J)
            CALL OUT (A,LTN,NW)
GO TO 140
120          CALL IN (A,LTN,NW)
C NEW TEMPERATURE KNOWN - SET UP T(OLD):
C
NPTS = 2*N2
NSKIP = 1
MTRN = N1H
MSKIP = N2P
ISIGN = 1
LOG = LOG2 (NPTS)
IEXSHFT = NMAX2/NPTS
CALL ACAC (A,EX,WORK)
DO 130 J = 1,N1H
    DO 130 I = 1,N2PD
130          A(I,J) = A(I,J)*.5
            CALL OUT (A,LTO,NW)
140          T = 0.
            DTT = DT
            NSTEPS = 1
145          CALL RUNINFO
C THIS IS THE BEGINNING OF THE TIME STEP.
C
150          IF ((MOD(NSTEPS, KTF) .EQ. 1) .AND. (NOTEMP .NE. 1))
1                  CALL TEMPFOU (CA,N2P,N1H,CWORK)
1          IF ((MOD(NSTEPS, KTF) .EQ. 1) .AND. (NOSALT .NE. 1))
1                  CALL SALTFOU (CA,N2P,N1H,CWORK)
IF (MOD(NSTEPS, KPF) .EQ. 1) CALL PSIFOU (CA,N2P,N1H,CWORK)
IF (MOD(NSTEPS, KZF) .EQ. 1) CALL ZETAFOU (CA,N2P,N1H,CWORK)
IF (MOD(NSTEPS, KVF) .EQ. 1) CALL VVELFOU (CA,N2P,N1H,FX,CWORK)
IF (MOD (NSTEPS+1,KTA) .NE. 1) GO TO 211
C
C TIME AVERAGE VORTICITY:
C
DTT = DT
T = T-.5*DT
CALL IN (A,LZO,NW)
NPTS = 2*N2
NSKIP = 1
MTRN = N1H
MSKIP = N2P
ISIGN = -1
IEXSHFT = NMAX2/NPTS
LOG = LOG2 (NPTS)
CALL ACAC (A,EX,WORK)
F = .125/FLOAT(N2)
DO 160 J = 1,N1H
    DO 160 I = 1,N2PD
160          A(I,J) = A(I,J)*F
            CALL OUT (A,LZO,NW)
ISIGN = 1
MTRN = N1Q
KZN = LZN
KZO = LZO

```

```

DO 180 K = 1,2
  CALL IN (A,KZO,NWH)
  CALL IN (A(1,K2Q),KZN,NWH)
  DO 170 J = 1,N1Q
    K2QJ = J+K2QM
    DO 170 I = 1,N2PD

```

Jul 13 18:46 1984 dbldif Page 8

```

170          A(I,J) = A(I,J)+0.5*A(I,K2QJ)
  CALL OUT (A,KZN,NWH)
  CALL ACAC (A,EX,WORK)
  DO 175 J = 1,N1Q
    DO 175 I = 1,N2PD
      A(I,J) = 0.5*A(I,J)
  CALL OUT (A,KZO,NWH)
  KZO = KZO+NWH
180          KZN = KZN+NWH
  IF (NOSALT .EQ. 1) GO TO 2211
C
C TIME AVERAGE SALT:
C
  ISIGN = -1
  MTRN = N1H
  CALL IN (A,LSO,NW)
  CALL ACAC (A,EX,WORK)
  F = 0.125/FLOAT(N2)
  DO 1190 J = 1,N1H
    DO 1190 I = 1,N2PD
      A(I,J) = F*A(I,J)
  CALL OUT (A,LSO,NW)
  ISIGN = 1
  MTRN = N1Q
  KTO = LSO
  KTN = LSN
  DO 2210 K = 1,2
    CALL IN (A,KTO,NWH)
    CALL IN (A(1,K2Q),KTN,NWH)
    DO 2200 J = 1,N1Q
      K2QJ = J+K2QM
      DO 2200 I = 1,N2PD
        A(I,J) = A(I,J)+0.5*A(I,K2QJ)
  CALL OUT (A,KTN,NWH)
  CALL ACAC (A,EX,WORK)
  DO 2205 J = 1,N1Q
    DO 2205 I = 1,N2PD
      A(I,J) = 0.5*A(I,J)
  CALL OUT (A,KTO,NWH)
  KTO = KTO+NWH
2210          KTN = KTN+NWH
2211          IF (NOTEMP .EQ. 1) GO TO 211
C
C TIME AVERAGE TEMPERATURE:
C
  ISIGN = -1
  MTRN = N1H
  CALL IN (A,LTO,NW)
  CALL ACAC (A,EX,WORK)
  F = 0.125/FLOAT(N2)
  DO 190 J = 1,N1H
    DO 190 I = 1,N2PD
      A(I,J) = F*A(I,J)
  CALL OUT (A,LTO,NW)
  ISIGN = 1
  MTRN = N1Q

```

```

KTO = LTO
KTN = LTN
DO 210 K = 1,2
    CALL IN (A,KTO,NWH)
    CALL IN (A(1,K2Q),KTN,NWH)
    DO 200 J = 1,N1Q
        K2QJ = J+K2QM

Jul 13 18:46 1984 dbldif Page 9

      DO 200 I = 1,N2PD
      A(I,J) = A(I,J)+0.5*A(I,K2QJ)
      CALL OUT (A,KTN,NWH)
      CALL ACAC (A,EX,WORK)
      DO 205 J = 1,N1Q
          DO 205 I = 1,N2PD
              A(I,J) = 0.5*A(I,J)
              CALL OUT (A,KTO,NWH)
              KTO = KTO+NWH
      210           KTN = KTN+NWH
C
C THE BEGINNING OF A NORMAL TIME STEP
C (U IS X VELOCITY AND V IS Y VELOCITY)
C FIND (-U) = D(PSI)/DY
C
211   NPTS = 2*N2
NSKIP = 1
MTRN = N1H
MSKIP = N2P
ISIGN = 1
LOG = LOG2 (NPTS)
IEXSHFT = NMAX2/NPTS
CALL IN (CA,LPN,NW)
F = 0.
DO 230 I = 1,N2
    DO 220 J = 1,N1H
        CA(I,J) = CMPLX(-F*AIMAG(CA(I,J)), F*REAL(CA(I,J)))
220
230   F = F+1.
DO 235 J = 1,N1H
    CA(N2P,J) = 0.
    CALL SCSC (CA,EX,WORK)
    CALL OUT (CA,LPY,NW)
    IF (KSM .EQ. 1) CALL MEAN (CA,U,G)
    IF (MOD(NSTEPS, KUF) .EQ. 1) CALL UVELFOU (CA,N2P,N1H,G,EX,CWORK)
    CALL IN (CA,LPN,NW)
C
C FIND V = D(PSI)/DX
C
    F = 0.
    DO 250 J = 1,N1H
        DO 240 I = 1,N2P
            CA(I,J) = CMPLX(-F*AIMAG(CA(I,J)), F*REAL(CA(I,J)))
240
250   F = F+FX
    CALL ACAC (CA,EX,WORK)
    CALL OUT (CA,LPN,NW)
    IF (MOD (NSTEPS, KPP) .EQ. 1) CALL PSIPHY (CA,N2P,N1H,EX,G,U,
                                                WORK,CWORK3)
1   IF ((MOD(NSTEPS, KPP) .EQ. 1) .AND. (NOTEMP .NE. 1))
1   CALL IN (CA,LPN,NW)

NPTS = 2*N2
NSKIP = 1
MTRN = N1H
MSKIP = N2P
ISIGN = 1
LOG = LOG2(NPTS)

```

```

IEXSHFT = NMAX2/NPTS
IF (NOSALT .EQ. 1) GO TO 2271
C
C T(TOTAL) = T-BETA1*Y
C WHERE BETA1 < Ø MEANS STABLE STRATIFICATION AND
C BETA1 > Ø MEANS UNSTABLE STRATIFICATION
C
C SAME FOR SALT STRATIFICATION ONLY WITH BETA2

Jul 13 18:46 1984 dbldif Page 10

      KTO = LSO
      DTB = Ø.5*BETA2*DTT
      DO 227Ø J = 1,N1H
         CALL IN (CWORK,KTO,N2PD)
         F = -FLOAT(J-1)*FX*DTT*Ø.25
         DO 226Ø I = 1,N2P
            CWORK(I) = CWORK(I)*CMPLX(1., U(I)*F)+DTB*CA(I,J)
      226Ø      CALL OUT (CWORK,KTO,N2PD)
      227Ø      KTO = KTO+N2PD
      2271      IF (NOTEMP .EQ. 1) GO TO 271
                 KTO = LTO
                 DTB = Ø.5*BETA1*DTT
                 DO 27Ø J = 1,N1H
                    CALL IN (CWORK,KTO,N2PD)
                    F = -FLOAT(J-1)*FX*DTT*Ø.25
                    DO 26Ø I = 1,N2P
                       CWORK(I) = CWORK(I)*CMPLX(1., U(I)*F)+DTB*CA(I,J)
      26Ø      CALL OUT (CWORK,KTO,N2PD)
      27Ø      KTO = KTO+N2PD
      271      CALL IN (CA,LZN,NW)
C
C FIND DZ/DY:
C
      F = Ø.
      DO 29Ø I = 1,N2
         DO 28Ø J = 1,N1H
            CA(I,J) = CMPLX(-F*AIMAG(CA(I,J)), F*REAL(CA(I,J)))
      28Ø      F = F+1.
      29Ø      DO 295 J = 1,N1H
                 CA(N2P,J) = Ø.
                 CALL SCSC (CA,EX,WORK)
                 CALL OUT (CA,LZY,NW)
                 CALL IN (CA,LZN,NW)
                 CALL ACAC (CA,EX,WORK)
                 DO 30Ø J = 1,N1H
                    DO 30Ø I = 1,N2P
                       CA(I,J) = CA(I,J)*Ø.5
      30Ø      CALL OUT (CA,LZN,NW)
      NPTS = N1
      NSKIP = N2P
      MTRN = N2P
      MSKIP = 1
      ISIGN = 1
      LOG = LOG2 (NPTS)
      IEXSHFT = NMAX2/NPTS
      IF (MOD(NSTEPS, KZP) .EQ. 1) CALL ZETAPHY (CA,N2PD,N1H,EX,WORK)
      IF (MOD(NSTEPS, KZP) .EQ. 1) CALL IN (CA,LZN,NW)
C
C FIND DZ/DX:
C
      F = Ø.
      DO 32Ø J = 1,N1H
         DO 31Ø I = 1,N2P
            CA(I,J) = CMPLX(-F*AIMAG(CA(I,J)), F*REAL(CA(I,J)))
      31Ø

```

```

320      F = F+FX2
CALL CSR (CA,EX)
CALL OUT (CA,LZX,NW)
CALL IN (CA,LZY,NW)
CALL CSR (CA,EX)
CALL OUT (CA,LZY,NW)
CALL IN (CA,LPN,NW)
CALL CSR (CA,EX)
CALL OUT (CA,LPN,NW)

```

Jul 13 18:46 1984 dbldif Page 11

```

CALL IN (CA,LPY,NW)
CALL CSR (CA,EX)
CALL OUT (CA,LPY,NW)
C
C WE ARE IN PHYSICAL SPACE NOW - FIND (U,V).GRAD (Z):
C
IF (MOD(NSTEPS, KUP) .EQ. 1) CALL UVELPHY (A,N2PD,N1H,WORK)
IF (MOD(NSTEPS, KVP) .EQ. 1) CALL VVELPHY (A,N2PD,N1H,WORK)
KPY = LPY
KPN = LPN
KZY = LZY
KZX = LZX
UKSM1F = FLOAT(2-KSM)
DO 340 K = 1,4
    CALL IN (A,KPY,NWQ)
    CALL IN (A(1,K1Q),KPN,NWQ)
    CALL IN (A(1,K2Q),KZY,NWQ)
    CALL IN (A(1,K3Q),KZX,NWQ)
    DO 330 I = 1,N2PD
        IPH = (I+1)/2
        GG = G(IPH)
        UU = U(IPH)*UKSM1F
        DO 330 J = 1,N1E
            A(I,J) = A(I,J)*GG + UU
            A(I,K2QM+J) = A(I,J)*A(I,K3QM+J) -
                A(I,K1QM+J)*A(I,K2QM+J)*GG
1           CONTINUE
330     CALL OUT (A,KPY,NWQ)
        CALL OUT (A(1,K2Q),KZY,NWQ)
        KPN = KPN+NWQ
        KPY = KPY+NWQ
        KZX = KZX+NWQ
        KZY = KZY+NWQ
340
C [Z(OLD) + (MEAN -U) * (-DZ(OLD)/DX)] + (U,V).GRAD(Z)
C
ISIGN = -1
CALL IN (A,LZY,NW)
CALL RCS (A,EX)
CALL OUT (A,LZY,NW)
KZO = LZO
KZN = LZN
KZY = LZY
DTFAC = 0.125*DTT/FLOAT(N1)
F1 = DTT*FX*0.25
F = F1
DO 360 K = 1,2
    CALL IN (CA,KZO,NWH)
    CALL IN (CA(1,K2Q),KZN,NWH)
    CALL OUT (CA(1,K2Q),KZO,NWH)
    CALL IN (CA(1,K2Q),KZY,NWH)
    DO 350 J = 1,N1Q
        K2QJ = J+K2QM

```

```

      F = F-F1
      DO 350 I = 1,N2P
        CA(I,J) = CA(I,J)*CMPLX(1.,F*U(I))+DTFAC*CA(I,K2QJ)
      CALL OUT (CA,KZN,NWH)
      KZN = KZN+NWH
      KZO = KZO+NWH
      KZY = KZY+NWH
 360    IF (NOSALT .EQ. 1) GO TO 4441
C
C FIND DS/DY:

```

Jul 13 18:46 1984 dbldif Page 12

```

C
      NPTS = 2*N2
      NSKIP = 1
      MTRN = N1H
      MSKIP = N2P
      ISIGN = 1
      LOG = LOG2 (NPTS)
      IEXSHFT = NMAX2/NPTS
      CALL IN (CA,LSN,NW)
      F = 0.
      DO 3380 I = 1,N2
        DO 3370 J = 1,N1H
          CA(I,J)=CMPLX(-F*AIMAG(CA(I,J)), F*REAL(CA(I,J)))
 3370
 3380    F = F+1.
      DO 3385 J = 1,N1H
        CA(N2P,J) = 0.
      CALL SCSC (CA,EX,WORK)
      CALL OUT (CA,LZY,NW)
      CALL IN (CA,LSN,NW)
      CALL ACAC (CA,EX,WORK)
      CALL OUT (CA,LSN,NW)
      NPTS = N1
      NSKIP = N2P
      MTRN = N2P
      MSKIP = 1
      LOG = LOG2 (NPTS)
      IEXSHFT = NMAX2/NPTS
      IF(MOD(NSTEPS,KTP).EQ.1)CALL SALTPHY(A,N2PD,N1H,EX,WORK)

```

```

C SALT INFLUENCE:
C

```

```

      F = -FX
      KZN = LZN
      KZX = LZX
      KTN = LSN
      DTA = 0.5*ALPHA2*DTT
      DO 4400 K = 1,2
        CALL IN (CA,KZN,NWH)
        CALL IN (CA(1,K2Q),KTN,NWH)
        DO 3390 J = 1,N1Q
          K2QJ = J+K2QM
          F = F+FX
          DO 3390 I = 1,N2P
            CA(I,K2QJ) = CMPLX(-F*AIMAG(CA(I,K2QJ)),
              F*REAL(CA(I,K2QJ)))
 3390    CA(I,J) = CA(I,J)+DTA*CA(I,K2QJ)
        CALL OUT (CA,KZN,NWH)
        CALL OUT (CA(1,K2Q),KZX,NWH)
        KZX = KZX+NWH
        KTN = KTN+NWH
 4400    KZN = KZN+NWH
 4441    CALL IN (A,LZX,NW)

```

```

CALL CSR (A,EX)
CALL OUT (A,LZX,NW)
CALL IN (A,LZY,NW)
CALL CSR (A,EX)
CALL OUT (A,LZY,NW)
KPN = LPN
KPY = LPY
KZX = LZX
KZY = LZY
C
C IN PHYSICAL SPACE - FIND (U,V).GRAD(T):
Jul 13 18:46 1984 dbldif Page 13

C
DO 4420 K = 1,4
CALL IN (A,KPY,NWQ)
CALL IN (A(1,K1Q),KPN,NWQ)
CALL IN (A(1,K2Q),KZY,NWQ)
CALL IN (A(1,K3Q),KZX,NWQ)
DO 4410 I = 1,N2PD
IPH = (I+1)/2
GG = G(IPH)
DO 4410 J = 1,NIE
A(I,J) = A(I,J)*A(I,K3QM+J) -
A(I,K1QM+J)*A(I,K2QM+J)*GG
1
CONTINUE
CALL OUT (A,KZY,NWQ)
KPN = KPN+NWQ
KPY = KPY+NWQ
KZX = KZX+NWQ
KZY = KZY+NWQ
4420
ISIGN = -1
CALL IN (A,LZY,NW)
CALL RCS (A,EX)
CALL OUT (A,LZY,NW)
KTN = LSN
KTO = LSO
KZY = LZY
DO 4440 K = 1,2
CALL IN (CA,KTO,NWH)
CALL IN (A(1,K2Q),KTN,NWH)
DO 4425 J = K2Q,N1H
DO 4425 I = 1,N2PD
A(I,J) = 0.5*A(I,J)
CALL OUT (A(1,K2Q),KTO,NWH)
CALL IN (CA(1,K2Q),KZY,NWH)
DO 4430 J = 1,N1Q
K2QJ = J+K2QM
DO 4430 I = 1,N2P
CA(I,J) = CA(I,J)+DTFAC*CA(I,K2QJ)
4430
CALL OUT (CA,KTN,NWH)
KTN = KTN+NWH
KTO = KTO+NWH
KZY = KZY+NWH
4440
4411 IF (NOTEMP .EQ. 1) GO TO 441
C
C FIND DT/DY:
C
NPTS = 2*N2
NSKIP = 1
MTRN = N1H
MSKIP = N2P
ISIGN = 1
LOG = LOG2 (NPTS)

```

```

IEXSHFT = NMAX2/NPTS
CALL IN (CA,LTN,NW)
F = Ø.
DO 380 I = 1,N2
  DO 370 J = 1,N1H
    CA(I,J) = CMPLX(-F*AIMAG(CA(I,J)), F*REAL(CA(I,J)))
  F = F+1.
DO 385 J = 1,N1H
  CA(N2P,J) = Ø.
CALL SCSC (CA,EX,WORK)
CALL OUT (CA,LZY,NW)
CALL IN (CA,LTN,NW)

```

Jul 13 18:46 1984 db1dif Page 14

```

CALL ACAC (CA,EX,WORK)
CALL OUT (CA,LTN,NW)
NPTS = N1
NSKIP = N2P
MTRN = N2P
MSKIP = 1
LOG = LOG2 (NPTS)
IEXSHFT = NMAX2/NPTS
IF (MOD(NSTEPS, KTP) .EQ. 1) CALL TEMPHY (A,N2PD,N1H,EX,WORK)

```

C C TEMPERATURE INFLUENCE:
C

```

F = -FX
KZN = LZN
KZX = LZX
KTN = LTN
DTA = Ø.5*ALPHA1*DTT
DO 400 K = 1,2
  CALL IN (CA,KZN,NWH)
  CALL IN (CA(1,K2Q),KTN,NWH)
  DO 390 J = 1,N1Q
    K2QJ = J+K2QM
    F = F+FX
    DO 390 I = 1,N2P
      CA(I,K2QJ) = CMPLX(-F*AIMAG(CA(I,K2QJ)),
                            F*REAL(CA(I,K2QJ)))
      CA(I,J) = CA(I,J)+DTA*CA(I,K2QJ)
    CALL OUT (CA,KZN,NWH)
    CALL OUT (CA(1,K2Q),KZX,NWH)
    KZX = KZX+NWH
    KTN = KTN+NWH
    KZN = KZN+NWH
    CALL IN (A,LZX,NW)
    CALL CSR (A,EX)
    CALL OUT (A,LZX,NW)
    CALL IN (A,LZY,NW)
    CALL CSR (A,EX)
    CALL OUT (A,LZY,NW)
    KPN = LPN
    KPY = LPY
    KZX = LZX
    KZY = LZY

```

C C IN PHYSICAL SPACE - FIND (U,V).GRAD(T):
C

```

DO 420 K = 1,4
  CALL IN (A,KPY,NWQ)
  CALL IN (A(1,K1Q),KPN,NWQ)
  CALL IN (A(1,K2Q),KZY,NWQ)
  CALL IN (A(1,K3Q),KZX,NWQ)

```

```

DO 410 I = 1,N2PD
IPH = (I+1)/2
GG = G(IPH)
DO 410 J = 1,N1E
A(I,J) = A(I,J)*A(I,K3QM+J) -
A(I,K1QM+J)*A(I,K2QM+J)*GG
1
CONTINUE
CALL OUT (A,KZY,NWQ)
KPN = KPN+NWQ
KPY = KPY+NWQ
KZX = KZX+NWQ
420 KZY = KZY+NWQ
ISIGN = -1

Jul 13 18:46 1984 dbldif Page 15

CALL IN (A,LZY,NW)
CALL RCS (A,EX)
CALL OUT (A,LZY,NW)
KTN = LTN
KTO = LTO
KZY = LZY
DO 440 K = 1.2
CALL IN (CA,KTO,NWH)
CALL IN (A(1,K2Q),KTN,NWH)
DO 425 J = K2Q,N1H
DO 425 I = 1,N2PD
A(I,J) = .5*A(I,J)
425 CALL OUT (A(1,K2Q),KTO,NWH)
CALL IN (CA(1,K2Q),KZY,NWH)
DO 430 J = 1,N1Q
K2QJ = J+K2QM
DO 430 I = 1,N2P
CA(I,J) = CA(I,J)+DTFAC*CA(I,K2QJ)
430 CALL OUT (CA,KTN,NWH)
KTN = KTN+NWH
KTO = KTO+NWH
KZY = KZY+NWH
440 DTF = .25*DTT*FX
441 F = -DTF
KZN = LZN
KTN = LSN
DO 4460 K = 1,2
IF (NOSALT.NE. 1) CALL IN (CA,KTN,NWH)
CALL IN (CA(1,K2Q),KZN,NWH)
DO 4450 J = 1,N1Q
K2QJ = J+K2QM
F = F+DTF
IF (NOSALT.EQ.1) GO TO 4449
DO 4448 I = 1,N2P
WORK(I) = F*U(I)
CA(I,J) = CMPLX (1.0, -WORK(I)) * CA(I,J) /
1
CONTINUE
DO 4450 I = 1,N2P
WORK(I) = F*U(I)
CA(I,K2QJ) = CMPLX (1.0, -WORK(I)) * CA(I,K2QJ) /
1
CONTINUE
IF (NOSALT.NE. 1) CALL OUT (CA,KTN,NWH)
CALL OUT (CA(1,K2Q),KZN,NWH)
4450 KTN = KTN+NWH
KZN = KZN+NWH
4460 F = -DTF
KZN = LZN

```

```

KTN = LTN
DO 460 K = 1,2
  IF (NOTEMP .NE. 1) CALL IN (CA,KTN,NWH)
  CALL IN (CA(1,K2Q),KZN,NWH)
  DO 450 J = 1,N1Q
    K2QJ = J+K2QM
    F = F+DTF
    IF (NOTEMP.EQ.1) GO TO 449
    DO 448 I = 1,N2P
      WORK(I) = F*U(I)
      CA(I,J) = CMPLX (1.0, -WORK(I)) * CA(I,J) /
      1                               (1.0 + WORK(I)*WORK(I))
1
448   CONTINUE
449   DO 450 I = 1,N2P

```

Jul 13 18:46 1984 dbldif Page 16

```

        WORK(I) = F*U(I)
        CA(I,K2QJ) = CMPLX (1.0, -WORK(I)) * CA(I,K2QJ) /
        1                               (1.0 + WORK(I)*WORK(I))
1
450   CONTINUE
        IF (NOTEMP .NE. 1) CALL OUT (CA,KTN,NWH)
        KTN = KTN+NWH
460   KZN = KZN+NWH
        NPTS = 2*N2
        NSKIP = 1
        MTRN = N1H
        MSKIP = N2P
        ISIGN = -1
        IEXSHFT = NMAX2/NPTS
        LOG = LOG2 (NPTS)
        CALL IN (A,LZN,NW)
        CALL ACAC (A,EX,WORK)
        CALL OUT (A,LZN,NW)
        IF (NOSALT .EQ. 1) GO TO 4461
          CALL IN (A,LSN,NW)
          CALL ACAC (A,EX,WORK)
          CALL OUT (A,LSN,NW)
4461   IF (NOTEMP .EQ. 1) GO TO 461
          CALL IN (A,LTN,NW)
          CALL ACAC (A,EX,WORK)
          CALL OUT (A,LTN,NW)
461   KZN = LZN
        KPN = LPN
        F2 = 1./(4.*FLOAT(N2))
        F1 = RNU*DTT
C
C FIND PRESSURE NOW OR NEVER (IF REQUIRED...)
C
1   IF (MOD(NSTEPS,KPRESS) .EQ. 1) CALL PRESURE (LPY, LPN, LZY,
2   1     LPRESS, NWEIG, A, CA, WORK, CWORK, G, EX, EVAL, IR, IC, E,
2   2     FX, N2P, N2PD, NIH, LMAPP)
2   CALL IN (EVAL,LEIG,NWEIG)
LVECT = N2M
LSKIP = 2
NVECT = 2
NVSKIP = 1
FXSQ = FX*FX
C
C UPDATE VORTICITY AND FIND THE NEW STREAMFUNCTION:
C
DO 480 J = 1,NIH
  CALL IN (CWORK,KZN,N2PD)
  IF (LMAPP .NE. 1) CALL SOLVE2 (CWORK(2),E,N2M,IR,IC)
  F = FLOAT((J-1)*(J-1))*FXSQ

```

```

DO 470 I = 2,N2
  CWORK(I) = CWORK(I)*F2 / (1.0-F1*(EVAL(I-1)-F))
  CWORK2(I) = CWORK(I) / (EVAL(I-1)-F)
470  CONTINUE
  IF (LMAPP .NE. 1) CALL LUMULT2 (CWORK2(2),E,N2M,IR,IC)
  IF (LMAPP .NE. 1) CALL LUMULT2 (CWORK(2),E,N2M,IR,IC)
  CWORK(1) = CWORK(1)*F2/(1.0+F1*F)
  CWORK2(1) = 0.0
  IF (F.NE.0.0) CWORK2(1) = -CWORK(1)/F
  CWORK(N2P) = 0.0
  CWORK2(N2P) = 0.0
  CALL OUT (CWORK,KZN,N2PD)
  CALL OUT (CWORK2,KPN,N2PD)
  KPN = KPN+N2PD
480  KZN = KZN+N2PD

```

Jul 13 18:46 1984 db1dif Page 17

```

C DE-ALIASE UPDATED FIELDS IF REQUESTED:
C
IF (NOALS.EQ.1) CALL DEALSE (CA,N2P,N1H,LZN)
IF (NOALS.EQ.1) CALL DEALSE (CA,N2P,N1H,LPN)
IF (NOSALT .EQ. 1) GO TO 5501
C UPDATE SALT:
C
IF (NOALS.EQ.1) CALL IN (EVAL,LEIG,NWEIG)
KTN = LSN
F1 = RKAPA2*DTT
DO 5500 J = 1,N1H
  CALL IN (CWORK,KTN,N2PD)
  IF (LMAPP .NE. 1) CALL SOLVE2 (CWORK(2),E,N2M,IR,IC)
  F = FLOAT((J-1)*(J-1))*FXSQ
  DO 490 I = 2,N2
    CWORK(I) = CWORK(I)*F2 / (1.0-F1*(EVAL(I-1)-F))
490  CONTINUE
  IF (LMAPP .NE. 1) CALL LUMULT2 (CWORK(2),E,N2M,IR,IC)
  CWORK(1) = CWORK(1)*F2 / (1.0+F1*F)
  CWORK(N2P) = 0.0
  CALL OUT (CWORK,KTN,N2PD)
5500  KTN = KTN+N2PD
C DE-ALIASE IF REQUESTED:
C
IF (NOALS.EQ.1) CALL DEALSE (CA,N2P,N1H,LSN)
C
5501  IF (NOTEMP .EQ. 1) GO TO 501
C UPDATE TEMPERATURE:
C
IF (NOALS.EQ.1) CALL IN (EVAL,LEIG,NWEIG)
KTN = LTN
F1 = RKAPA1*DTT
DO 500 J = 1,N1H
  CALL IN (CWORK,KTN,N2PD)
  IF (LMAPP .NE. 1) CALL SOLVE2 (CWORK(2),E,N2M,IR,IC)
  F = FLOAT((J-1)*(J-1))*FXSQ
  DO 498 I = 2,N2
    CWORK(I) = CWORK(I)*F2 / (1.0-F1*(EVAL(I-1)-F))
498  CONTINUE
  IF (LMAPP .NE. 1) CALL LUMULT2 (CWORK(2),E,N2M,IR,IC)
  CWORK(1) = CWORK(1)*F2 / (1.0+F1*F)
  CWORK(N2P) = 0.0
  CALL OUT (CWORK,KTN,N2PD)

```

```

500      KTN = KTN+N2PD
501      IF (NOALS.EQ.1) CALL DEALSE (CA,N2P,N1H,LTN)
      DTT = DT+DT
      T = T+DT
      NSTEPS = NSTEPS+1
      IF((KTAPE.EQ.2).OR.(KTAPE.EQ.3))CALL DATAOUT(A,WORK)
      IF (NUMBER.GE.NSTEPS) GOTO 150
      STOP FINE
      END
*****
      SUBROUTINE DEALSE (CA,N2P,N1H,LOC)
      COMPLEX CA(N2P,1)

C THIS FUNCTION GIVES THE INDEX OF THE HIGHEST MODE TO BE RETAINED
C IN TERMS OF THE NUMBER OF MODES WHICH CAN BE RESOLVED:
C

Jul 13 18:46 1984 db1af Page 18

      LASTMOD (MODEHI) = IFIX (0.666666666666666*FLOAT(MODEHI-1)+0.999)
C
      NW = 2*N2P*N1H
      CALL IN (CA,LOC,NW)
      N1CUT = LASTMOD (N1H) +1
      N2CUT = LASTMOD (N2P) +1
C
C THIS DEALIASING PROCEDURE HAS TO BE IMPROVED IF NONLINEAR MAPPING
C IN Y-DIRECTION IS USED:
C
      DO 10 J=N1CUT,N1H
      DO 10 I=1,N2P
10      CA(I,J) = 0.0
      DO 20 J=1,N1H
      DO 20 I=N2CUT,N2P
20      CA(I,J) = 0.0
      CALL OUT (CA,LOC,NW)
      RETURN
      END
*****
      SUBROUTINE START (NX,NY)
C
C THIS SUBROUTINE IS USED TO SPECIFY OPTIONS FOR PROGRAM OCEAN.
C
      COMMON /OPTIONS/ N1, N2, NUMBER, KDATA, LMAPP, NOTEMP, KTA, KTAPE,
1      KSM, KPP, KZP, KTP, KUP, KVP, KPF, KZF, KTF, KUF, KVF,
2      KPRESS, NOALS, NOSALT

C N1 AND N2 ARE THE DIMENSIONS OF THE MATRIX. THEY HAVE TO BE POWERS
C OF 2 AND GREATER THAN OR EQUAL TO 8
C
      N1 = NX
      N2 = NY
C
C KDATA SPECIFIES THE WAY DATA IS SUPPLIED TO THE PROGRAM.
C
      KDATA = 1      MEANS THAT IT WILL BE IN PHYSICAL SPACE
      = 2      MEANS THAT IT WILL BE IN FOURIER SPACE
      = 3      MEANS THAT THIS IS A RESTART RUN (->READ TAPE)
C
      KDATA = 1
C
C LMAPP SPECIFIES THE MAPPING IN THE Y DIRECTION.
C
      LMAPP = 1      MEANS LINEAR MAPPING (-> EIGSET NOT CALLED)
      = 2      MEANS NON-LINEAR MAPPING

```

C = 3 MEANS RESTART RUN, NON-LINEAR MAPPING

C LMAPP = 1

C NOTEMP EXCLUDES TEMPERATURE FIELD FROM CALCULATIONS WHEN SET TO 1.

C NOTEMP = 0

C NOSALT EXCLUDES SALT FIELD FROM CALCULATIONS WHEN SET TO 1.

C NOSALT = 0

C NOALS TURNS ON DEALIASING AT EACH TIME STEP WHEN SET TO 1:
(TO BE USED ONLY WITH LINEAR MAPPING)

C NOALS = 1

Jul 13 18:46 1984 dbldif Page 19

C C TIME AVERAGING IS PERFORMED EACH KTA TIME STEPS:

C KTA = 30

C C THE FOLLOWING PARAMETERS SPECIFY WHEN TO CALL CORRESPONDING OUTPUT
C SUBROUTINES. IF YOU WANT, FOR EXAMPLE, PSIPHY TO BE CALLED EVERY
C 50 TIME STEPS, YOU SHOULD SET KPP TO 50.

C C WARNING: SETTING A PARAMETER TO 1 TURNS OFF THE SUBROUTINE. IT DOES
C NOT CAUSE IT TO BE CALLED EVERY TIME STEP.
C YOU SHOULD NEVER SET ANY OF THEM TO 0, SINCE THEY ARE USED
C WITH THE FUNCTION MOD(NSTEPS,KXYZ).

C KPP = 1250

KZP = 625

KTP = 250

KUP = 125

KVP = 125

KPF = 1

KZF = 1

KTF = 1

KUF = 1

KVF = 1

KPRESS = 1

C KTAPE SPECIFIES HOW WILL HISTORY TAPE BE WRITTEN:

C KTAPE = 1 WRITE ONLY EIG-STUFF ON TAPE

= 2 WRITE ONLY DATA ON TAPE

= 3 WRITE BOTH EIG-STUFF AND DATA ON TAPE

C ANY OTHER VALUE WILL PREVENT THE HISTORY TAPE FROM BEING WRITTEN.

C KTAPE = 0

C NUMBER SPECIFIES THE NUMBER OF TIME STEPS IN A SINGLE RUN:

C NUMBER = 2501

C KSM GOVERNS THE USE OF THE MEAN FLOW OPTION.

C C KSM = 0 MEANS THAT U(MEAN) IS ZERO

1 MEANS THAT IT SHOULD BE COMPUTED

2 MEANS THAT IT IS USER-SPECIFIED

```

C      KSM = 1
C THAT WAS ALL. RETURN TO MOTHER OCEAN.
C
C      RETURN
C      END
*****
SUBROUTINE GSET (GMAPP)
DIMENSION GMAPP(1)
COMMON /PI/ PI
COMMON /NMBRS/ NSKIP(4), N2P
F = PI/FLOAT(N2P-1)
C
C GSET DEFINES THE MAPPING FUNCTION.
C
DO 20 I = 1,N2P
20   GMAPP(I) = F*FLOAT(I-1)

Jul 13 18:46 1984 dbldif Page 20

      RETURN
      END
*****
SUBROUTINE EIGSET (CA,G,GMAPP,CWORK,EX,EVAL,IR,IC,E,N2M)
COMMON /SOLV/ LVECT, LSKIP, NVECT, NVSKIP
COMMON /FT/ NPTS, NSKIP, MTRN, MSKIP, ISIGN, LOG, IEXSHFT
COMMON /EXSIZE/ NMAX2
COMMON /PI/ PI
COMMON /DATA/ SKIP(3), H
COMMON /LADDR/ LDUMMY(8), LEIG, LE, LH, LEV,
1LSN,LSO
DIMENSION INDIC(127)
DIMENSION G(1),GMAPP(1),EVAL(1)
DIMENSION IR(1),IC(1),E(N2M,1)
COMPLEX CA(1),EX(1),CWORK(1)

C
N2 = N2M+1
N2P = N2+1
H = GMAPP(N2P)
F = H/FLOAT(N2)
DO 10 I = 1,N2P
10   G(I) = GMAPP(I)-F*FLOAT(I-1)
     CA(I) = CMPLX (G(I), 0.)
CA(1) = 0.
CA(N2P) = 0.
NPTS = 2*(N2P-1)
NSKIP = 1
MTRN = 1
MSKIP = 1
IEXSHFT = NMAX2/NPTS
LOG = LOG2 (NPTS)
ISIGN = -1
CALL ACAC (CA,EX,CWORK)
DO 20 I = 1,N2P
20   CA(I) = CA(I)*CMPLX(0., FLOAT(I-1))
CA(N2P) = 0.
ISIGN = 1
CALL SCSC (CA,EX,CWORK)
F = 8.*FLOAT(N2P-1)*H/PI
FN = 8.*FLOAT(N2P-1)
DO 30 I = 1,N2P
30   G(I) = FN/(F+REAL(CA(I)))
NWM = N2M*N2M
NWEIG = N2M*(N2M+3)

```

```

AFACT = 2./FLOAT(N2)
PIFACT = PI/FLOAT(N2)
NPTS = 2*N2
NSKIP = 1
MTRN = 1
MSKIP = 1
LOG = LOG2 (NPTS)
IEXSHFT = NMAX2/NPTS
FF = 1./(16.*FLOAT(N2*N2))
KE = LE
DO 100 J = 1,N2M
    FACT = PIFACT*FLOAT(J)
    DO 40 I = 1,N2P
        CWORK(I) = CMPLX(0.,G(I)*FLOAT(J)*COS(FACT*FLOAT(I-1)))
40     ISIGN = -1
        CALL SCSC (CWORK,EX,E)
        DO 50 I = 1,N2
            CWORK(I) = CWORK(I)*CMPLX(0.,FLOAT(I-1))
50     CWORK(N2P) = 0.

```

Jul 13 18:46 1984 dbldif Page 21

```

ISIGN = 1
CALL ACAC (CWORK,EX,E)
DO 60 I = 1,N2P
    CWORK(I) = G(I)*CWORK(I)
60     ISIGN = -1
        CALL ACAC(CWORK,EX,E)
        DO 70 I = 2,N2
            E(I-1,1) = REAL(CWORK(I))*FF
70     CALL OUT (E,KE,N2M)
100     KE = KE + N2M
        CALL IN (E,LE,NWM)
C
C COMPUTE EIGENVALUES AND EIGENVECTORS, CHECK SUCCESS AND STOP IF NOT 0.
C
    CALL LCMRQR (E,N2M,N2M,EVAL,1,LH,LEV,INDIC)
    LVECT = N2M
    LSKIP = 2
    NVECT = 1
    NVSKIP = 1
    DO 110 I = 1,N2M
        IF (INDIC(I) - 1) 1000, 1001, 110
1000     PRINT 210, I,INDIC(I)
        STOP EIGS

```

C IF AN EIGENVECTOR WAS NOT FOUND, COMPUTE IT THE OLD WAY

```

C
1001     CALL OUT (E,LEV,NWM)
    CALL IN (E,LE,NWM)
    DO 1010 J = 1,N2M
        CWORK(J) = 1.
1010     E(J,J) = E(J,J)-EVAL(I)
        CALL LU2 (E,N2M,N2M,IR,IC)
        DO 1020 K = 1,3
            CALL SOLVE2 (CWORK,E,N2M,IR,IC)
            SUM = 0.
        DO 1030 J = 1,N2M
            SUM = SUM + REAL(CWORK(J))*REAL(CWORK(J))
1030     IF (SUM .EQ. 0.) GO TO 1020
            F = 1./SQRT(SUM)
            DO 1040 J = 1,N2M
                CWORK(J) = REAL(CWORK(J))*F
1040     CONTINUE
1020     XLARGE = 0.

```

```

DO 1060 J = 1,N2M
  IF (ABS(REAL(CWORK(J))) .LT. XLARGE) GO TO 1060
  XLARGE = ABS(REAL(CWORK(J)))
  IXL = J
CONTINUE
FACT = 1./REAL(CWORK(IXL))
CALL IN (E,LEV,NWM)
DO 1050 J = 1,N2M
  E(J,I) = REAL(CWORK(J))*FACT
  PRINT 220, I,INDIC(I),EVAL(I)
CONTINUE
110 FORMAT (* EIGSET ERROR, INDIC(*,I3,*),*,I3)
210 FORMAT (///,* LCMRQR FAILURE, INDIC(*,I3,*),*,I2,/,
1           * EIGENVECTOR CORRESPONDING TO EIGENVALUE *,E15.7,
2           * HAS BEEN COMPUTED THE OLD WAY*)

C LU TRANSFORM EIGENVECTORS AND STORE RESULTS IN LCM
C ASSUMING THAT EVAL, IR, IC AND E OCCUPY NWEIG CONSECUTIVE LOCATIONS
C CALL LU2 (E,N2M,N2M,IR,IC)

```

Jul 13 18:46 1984 db1dif Page 22

```

CALL OUT (EVAL,LEIG,NWEIG)
RETURN
END
*****SUBROUTINE LCMRQR(A,N,NDIM,EVAL,KS,LEV,LH,INDIC)
C
C LCMRQR FINDS THE N EIGENVALUES AND CORRESPONDING EIGENVECTORS
C OF THE GENERAL MATRIX A OF ORDER N
C
C THE QR METHOD OF FRANCIS IS USED TO COMPUTE THE EIGENVALUES
C AND INVERSE ITERATION IS USED TO COMPUTE THE EIGENVECTORS
C
C LARGE CORE MEMORY IS USED AS A TEMPORARY STOREGE PLACE
C THE PROGRAM REQUIRES 2 * (N * N) WORDS OF LCM
C
C A IS THE INPUT MATRIX (DESTROYED BY THE SUBROUTINE)
C N IS THE ORDER OF A
C NDIM IS THE DIMENSION OF A IN THE CALLING PROGRAM
C (N MUST BE LESS THAN THE MINIMUM OF NDIM AND 200)
C EVAL(J) WILL CONTAIN THE JTH EIGENVALUE
C RESULTING EIGENVECTORS ARE RETURNED IN A. A(I,J) WILL
C CONTAIN THE ITH COMPONENT CORRESPONDING TO THE JTH EIGENVALUE
C KS IS AN OPTION INDICATOR
C KS=0, COMPUTE EIGENVALUES ONLY
C KS=1, COMPUTE EIGENVALUES AND EIGENVECTORS
C LEV IS STARTING LCM LOCATION FOR N * N WORDS OF MEMORY
C LH IS ANOTHER SUCH LOCATION (THESE TWO BLOCKS SHOULD BE DISJUNCT)
C (IF KS=0, LEV AND LH CAN BE DUMMY VARIABLES)
C INDIC INDICATES THE SUCCESS OF LCMRQR AS FOLLOWS:
C
C VALUE OF INDIC(K)      EIGENVALUE(K)      EIGENVECTOR(K)
C
C   0          NOT FOUND        NOT FOUND
C   1          FOUND          NOT FOUND
C   2          FOUND          FOUND
C
C DIMENSION A(NDIM,1)
C DIMENSION EVAL(1),INDIC(1)
C
C COMMON BLOCK /QR/ CAN BE USED OUTSIDE LCMRQR AS A WORKSPACE
C COMMON /QR/ ALPHA(200), DUMMY(200), WORK(200),

```

```

1           SCALE(200), SUBDIA(200), D(200)
C
C   EPS IS A MACHINE DEPENDENT CONSTANT DEFINED AS FOLLOWS -
C
C   EPS=2.**(1-T) WHERE T IS THE NUMBER OF BINARY DIGITS IN THE
C   MANTISSA OF A SINGLE PRECISION FLOATING POINT NUMBER
C
C   EPS = 2.0**(-48)
C
C   CHECK THE ORDER OF THE MATRIX
C
C   IF(N.NE.1) GO TO 10
C
C   THE MATRIX IS OF ORDER 1
C
C   EVAL(1)=A(1,1)
C   INDIC(1)=1
C   IF(KS.EQ.0) RETURN
C   A(1,1)=1.
C   INDIC(1)=2
C   RETURN

```

Jul 13 18:46 1984 dbldif Page 23

```

C
C   CONDITION THE MATRIX
C
10  MAXIT=1
    IF(KS.EQ.1) MAXIT=10
    CALL EQUIL(A,N,NDIM,MAXIT,SCALE,ANORM)
C
C   REDUCE THE CONDITIONED MATRIX TO UPPER HESSENBERG FORM H
C
    CALL HESS(A,N,NDIM,SUBDIA,D)
C
C   IF THE EIGENVECTORS ARE DESIRED, SAVE THE RESULTS
C   OF THE REDUCTION TO HESSENBERG FORM
C
    IF(KS.EQ.0) GO TO 30
    DO 20 K=1,N
    KLH=LH+N*(K-1)
    CALL OUT (A(1,K),KLH,N)
20  CONTINUE
30  NM1=N-1
    DO 40 I=1,NM1
40  A(I+1,I)=SUBDIA(I)
C
C   COMPUTE A CRITERION (EPSREL) FOR EFFECTIVELY ZERO SUBDIAGONAL ELEM
C
    HNORM=0.
    DO 50 I=1,N
    LIM=I-1
    IF(LIM.EQ.0) LIM=1
    DO 50 J=LIM,N
50  HNORM=HNORM+A(I,J)*A(I,J)
    HNORM=SQRT(HNORM)
    EPSREL=EPS*HNORM
C
C   COMPUTE THE EIGENVALUES OF H
C
    CALL QR1(A,N,NDIM,EPSREL,EVAL,INDIC)
    IF(KS.EQ.0) GO TO 220
C
C   COMPUTE THE EIGENVECTORS OF H
C

```

```

C      CHECK FOR ZERO SUBDIAGONAL ELEMENTS TO REDUCE THE
C      AMOUNT OF WORK TO FIND THE EIGENVECTORS
C
C      NSQ=N*N
C      NF=N
60    NS=1
      IF(NF.EQ.1) GO TO 80
      DO 70 IBACK=2,NF
      I=NF-IBACK+2
      IF(ABS(SUBDIA(I-1)).GT.EPSREL) GO TO 70
      NS=I
      GO TO 80
70    CONTINUE
80    DO 120 K=NS,NF
      IF(INDIC(K).EQ.0) GO TO 120
      EV=EVAL(K)
      IF(K.EQ.NS) GO TO 100
C
C      SMALL PERTURBATION OF EQUAL EIGENVALUES TO
C      OBTAIN A COMPLETE SET OF EIGENVECTORS
C
      R=0.

```

Jul 13 18:46 1984 dbldif Page 24

```

KM1=K-1
DO 90 I=NS,KM1
TEMP=EV-EVAL(I)
IF(TEMP.NE.0.) GO TO 90
R=R+3.
90 CONTINUE
IF(R.EQ.0.) GO TO 100
EV=EV+R*EPS*AMAX1{1.,ABS(EV)}
C
C      COMPUTE THE EIGENVECTOR CORRESPONDING TO EV
C
100 CALL IN (A,LH,NSQ)
CALL VECTR(A,NF,N,SUBDIA,EV,WORK,INDIC(K))
C
C      AUGMENT THE VECTOR WITH ZEROS IN POSITIONS NF+1,...,N
C
IF(NF.EQ.N) GO TO 115
NFP1=NF+1
DO 110 I=NFP1,N
WORK(I)=0.
115 KLEV=LEV+N*(K-1)
CALL OUT (WORK,KLEV,N)
120 CONTINUE
IF(NS.EQ.1) GO TO 130
NF=NS-1
GO TO 60
C
C      TRANSFORM THE EIGENVECTORS FROM EIGENVECTORS OF H
C      TO EIGENVECTORS OF THE CONDITIONED MATRIX
C
130 DO 135 K=1,N
      KLEV=LEV+N*(K-1)
      CALL IN (A(1,K),KLEV,N)
135 CONTINUE
      DO 180 KBACK=1,NM1
      K=NM1-KBACK+1
      KP1=K+1
      IF(K.EQ.NM1) GO TO 160
C
C      COMPUTE THE ROW VECTOR ALPHA=TRANSPOSE-W(K)*EVECT

```

```

C
KLH=LH+N*(K-1)
CALL IN (WORK,KLH,N)
DO 140 J=1,N
ALPHA(J)=0.
DO 140 I=KP1,N
140 ALPHA(J)=ALPHA(J)+WORK(I)*A(I,J)
C
C COMPUTE EVECT=P(K)*EVECT=EVECT-W*ALPHA
C
DO 150 J=1,N
DO 150 I=KP1,N
150 A(I,J)=A(I,J)-WORK(I)*ALPHA(J)
160 DO 170 J=1,N
170 A(KP1,J)=D(K)*A(KP1,J)
180 CONTINUE
C
C COMPUTE THE EIGENVECTORS OF THE ORIGINAL MATRIX
C ( I. E. REMOVE THE SCALING FACTORS)
C
DO 190 I=1,N
DO 190 J=1,N
190 A(I,J)=A(I,J)/SCALE(I)

```

Jul 13 18:46 1984 dbldif Page 25

```

C
C NORMALIZE THE EIGENVECTORS
C
DO 210 J=1,N
BIG=0.
DO 200 I=1,N
RS=A(I,J)*A(I,J)
IF(BIG.GE.RS) GO TO 200
IBIG=I
BIG=RS
200 CONTINUE
TEMP=A(IBIG,J)
DO 210 I=1,N
210 A(I,J)=A(I,J)/TEMP
C
C COMPUTE THE EIGENVALUES OF THE ORIGINAL MATRIX
C ( I. E. REMOVE THE NORMALIZATION FACTOR)
C
220 DO 230 I=1,N
230 EVAL(I)=EVAL(I)*ANORM
RETURN
END
*****
SUBROUTINE EQUIL(A,N,NDIM,MAXIT,SCALE,ANORM)
C
C THE MATRIX A IS BALANCED BY PERFORMING DIAGONAL SIMILARITY
C TRANSFORMATIONS ON A SO THAT CORRESPONDING ROWS AND COLUMNS
C HAVE APPROXIMATELY EQUAL NORMS
C THE MATRIX IS THEN NORMALIZED SO THAT THE LARGEST ELEMENT
C OF THE MATRIX IS APPROXIMATELY 1.0 IN MAGNITUDE
C
C THE DIAGONAL SCALING CONSTANTS ARE STORED IN THE LINEAR ARRAY
C SCALE AND THE NORMALIZING FACTOR IS STORED IN ANORM
C FOR SUBSEQUENT USE IN EIGENVECTOR CALCULATIONS
C
C REFERENCE-ON PRE-CONDITIONING OF MATRICES. E. E. OSBORNE
C ACM JOURNAL,7,(4),PGS.338-345 (1960)
C
DIMENSION A(NDIM,1),SCALE(1)

```

```

C      INITIALIZATION OF VARIABLES
C
C      ITER=0
C      ELN16=ALOG(16.0)
C      DO 10 I=1,N
C 10  SCALE(I)=1.

C      BALANCE CONTROL LOOP

C 20 NCOUNT=0
C      DO 70 I=1,N

C      CALCULATE THE EUCLIDEAN NORM OF THE ITH ROW AND COLUMN
C      EXCLUDING THE DIAGONAL ELEMENT (SINCE IT WILL NOT BE
C      ALTERED BY THE SIMILARITY TRANSFORMATION)

C      ROW=0.
C      COLUMN=0.
C      DO 30 J=1,N
C      IF(I.EQ.J) GO TO 30
C      ROW=ROW+A(I,J)*A(I,J)
C      COLUMN=COLUMN+A(J,I)*A(J,I)
C 30 CONTINUE

```

Jul 13 18:46 1984 dbldif Page 26

```

C      CHECK FOR ZERO ROW OR COLUMN NORM
C
C      IF(ROW.EQ.0.) GO TO 40
C      IF(COLUMN.EQ.0.) GO TO 40

C      EQUALIZE (APPROXIMATELY) THE NORMS OF THE ITH ROW AND COLUMN

C      REXP=ALOG(COLUMN/ROW)/ELN16
C      IF(REXP.LT.0.) IEXP=REXP-.5
C      IF(REXP.GE.0.) IEXP=REXP+.5
C      IF(IEXP.NE.0) GO TO 50

C      IF THE ITH ROW AND COLUMN ARE ALREADY IN BALANCE, INCREMENT NCOUNT

C 40 NCOUNT=NCOUNT+1
C      GO TO 70

C      DIAGONAL SCALING FACTOR IS COMPUTED AS A POWER OF TWO SO THAT
C      NO ERROR WILL BE INTRODUCED (I.E. ONLY THE EXPONENTS WILL CHANGE)

C 50 FACTOR=2.**IEXP
C      DO 60 J=1,N
C      IF(I.EQ.J) GO TO 60
C      A(I,J)=A(I,J)*FACTOR
C      A(J,I)=A(J,I)/FACTOR
C 60 CONTINUE

C      ACCUMULATE THE SCALING CONSTANT

C      SCALE(I)=SCALE(I)*FACTOR
C 70 CONTINUE

C      INCREMENT THE ITERATION COUNTER AND CHECK THE NUMBER OF ITERATIONS

C      ITER=ITER+1
C      IF(ITER.EQ.MAXIT) GO TO 80

```

```

C IF NCOUNT IS NOT EQUAL TO N, MAKE ANOTHER PASS
C IF(NCOUNT.LT.N) GO TO 20
C NORMALIZATION OF A
C SEARCH FOR THE LARGEST ELEMENT (IN MAGNITUDE) OF A
80 ANORM=0.
DO 90 I=1,N
DO 90 J=1,N
90 ANORM=AMAX1(ABS(A(I,J)),ANORM)
IF(ANORM.EQ.0.) RETURN
C NORMALIZATION FACTOR IS COMPUTED AS A POWER OF TWO SO THAT
NO ERROR WILL BE INTRODUCED
REXP=ALOG(ANORM)/ALOG(2.)
IF(REXP.LT.0.) IEXP=REXP-.5
IF(REXP.GE.0.) IEXP=REXP+.5
ANORM=2.**IEXP
DO 100 I=1,N
DO 100 J=1,N
100 A(I,J)=A(I,J)/ANORM
RETURN

```

Jul 13 18:46 1984 dbldif Page 27

```

END
C*****SUBROUTINE HESS(A,N,NDIM,Subdia,D)
C HOUSEHOLDER REDUCTION OF THE COMPLEX MATRIX A TO UPPER HESSENBERG
C THE UPPER HALF OF A WILL CONTAIN THE UPPER HALF OF H
C THE LOWER HALF OF A WILL CONTAIN THE SPECIAL VECTORS USED IN
C THE DEFINITION OF THE HOUSEHOLDER TRANSFORMATION MATRICES
C THE LINEAR ARRAY SUBDIA WILL CONTAIN THE SUBDIAGONAL ELEMENTS OF H
C REFERENCE-THE ALGEBRAIC EIGENVALUE PROBLEM,WILKINSON,PGS.347-351
C REFERENCE-NUMERISCHE MATHEMATIK,VOL.8,1966,PGS.72-89
C DIMENSION A(NDIM,1),SUBDIA(1),D(I)
COMMON /QR/ WVEC(200)
C EPSSQ IS A MACHINE DEPENDENT CONSTANT DEFINED AS FOLLOWS-
EPSSQ=EPS*EPS
C EPSSQ = 4.0***(1-48)
C REDUCTION LOOP
C NM1=N-1
DO 120 K=1,NM1
KP1=K+1
KP2=K+2
C INSURE THAT THE SUBDIAGONAL ELEMENT IS REAL BY PERFORMING A
UNITARY DIAGONAL SIMILARITY TRANSFORMATION TO TRANSFORM THE
ELEMENT TO ITS MODULUS
C THIS TRANSFORMATION GREATLY SIMPLIFIES THE HOUSEHOLDER REDUCTION
AND ALSO FULFILLS THE QR1 REQUIREMENT THAT THE SUBDIAGONAL
ELEMENTS OF THE HESSENBERG MATRIX H MUST BE REAL
C (SAVE THE TRANSFORMATION CONSTANT IN D FOR SUBSEQUENT USE
IN EIGENVECTOR CALCULATIONS)

```

```

C
D(K)=1.
R=ABS(A(KP1,K))
IF(R.EQ.0.) GO TO 30
D(K)=R/A(KP1,K)
DO 10 J=K,N
10 A(KP1,J)=A(KP1,J)*D(K)
DO 20 I=1,N
20 A(I,KP1)=A(I,KP1)/D(K)
30 IF(K.EQ.NM1) GO TO 130
C
C COMPUTE S**2
C
SS=0.
DO 40 I=KP2,N
40 SS=SS+A(I,K)*A(I,K)
C
C CHECK IF THE TRANSFORMATION IS TO BE SKIPPED
C (EPSSQ IS USED AS A CRITERION TO DETERMINE WHEN THE TRANSFORMATION
C CAN BE SKIPPED. THIS IS A RELATIVE TOLERANCE SINCE THE MAXIMUM
C ELEMENT OF A IS ABOUT 1.0 IN MAGNITUDE)
C
IF(SS.GT.EPSSQ) GO TO 60
SUBDIA(K)=A(KP1,K)
DO 50 I=KP1,N
50 A(I,K)=0.

```

Jul 13 13:46 1984 dbldif Page 28

```

GO TO 120
60 TEMP=A(KP1,K)
SS=SS+TEMP**2
C
C COMPUTE S AND THE NEW SUBDIAGONAL ELEMENT
C
S=SQRT(SS)
SUBDIA(K)=-S
C
C COMPUTE THE W VECTOR
C
WVEC(KP1)=SQRT(1.+TEMP/S)
A(KP1,K)=WVEC(KP1)
WNORM=1./(S*WVEC(KP1))
DO 70 I=KP2,N
WVEC(I)=WNORM*A(I,K)
70 A(I,K)=WVEC(I)
C
C PRE-MULTIPLICATION BY THE UNITARY MATRIX P
C
DO 90 J=KP1,N
C
C COMPUTE JTH COMPONENT OF THE P VECTOR
C
P=0.
DO 80 I=KP1,N
80 P=P+WVEC(I)*A(I,J)
C
C COMPUTE THE JTH COLUMN OF F=P*A
C
DO 90 I=KP1,N
90 A(I,J)=A(I,J)-WVEC(I)*P
C
C POST-MULTIPLICATION BY THE UNITARY MATRIX P
C
DO 110 I=1,N

```

```

C COMPUTE ITH COMPONENT OF THE Q VECTOR
C
C Q=Ø.
C DO 100 J=KP1,N
100 Q=Q+A(I,J)*WVEC(J)
C COMPUTE THE ITH ROW OF F*P=P*A*P
C
C DO 110 J=KP1,N
110 A(I,J)=A(I,J)-Q*WVEC(J)
120 CONTINUE
130 SUBDIA(N-1)=A(N,N-1)
RETURN
END
C*****SUBROUTINE QR1(H,NSUB,NDIM,EPS,EVAL,INDIC)
C
C COMPLEX QR METHOD TO FIND THE EIGENVALUES OF THE
C COMPLEX UPPER HESSENBERG MATRIX H
C
C REFERENCE-THE COMPUTER JOURNAL( THE QR TRANSFORMATION,FRANCIS)
C PART 1-OCT.1961,PGS.265-271
C PART 2-JAN.1962,PGS.332-345
C REFERENCE-THE ALGEBRAIC EIGENVALUE PROBLEM,WILKINSON
C
C DIMENSION H(NDIM,1),EVAL(1),INDIC(1)

```

Jul 13 18:46 1984 db1dif Page 29

```

COMMON /QR/ NU(200),MU(200)
REAL NU,KAP,MU
N=NSUB
C
C CONTROL LOOP TO COMPUTE NEXT EIGENVALUE
C
10 IF(N.EQ.1) GO TO 50
NM1=N-1
NOGO=Ø
KOUNT=Ø
EØ=Ø.
C
C IF A ZERO SUBDIAGONAL ELEMENT IS FOUND IN ROW NO, THEN THE
C EIGENVALUES OF THE PRINCIPAL SUBMATRIX OF ORDER N-NQ+1 IN
C THE LOWER RIGHT HAND CORNER OF H ARE COMPUTED
C
20 NQ=1
DO 30 IBACK=2,N
I=N-IBACK+2
IF(ABS(H(I,I-1)).GT.EPS) GO TO 30
NQ=I
GO TO 40
30 CONTINUE
40 IF(NQ.NE.N) GO TO 60
C
C THE NTH DIAGONAL ELEMENT HAS CONVERGED OR N=1
C
50 EVAL(N)=H(N,N)
INDIC(N)=1
IF(N.EQ.1) RETURN
C
C DEFLATE THE MATRIX BY DECREMENTING N
C
N=NM1
GO TO 10

```

```

C THE NTH DIAGONAL ELEMENT HAS NOT CONVERGED
C COMPUTE THE EIGENVALUES OF THE PRINCIPAL 2 BY 2
C SUBMATRIX IN ROWS NM1 THROUGH N
C
60 DIF=.5*(H(NM1,NM1)-H(N,N))
DISC=DIF*DIF+H(NM1,N)*H(N,NM1)
C
IF(EIGENVALUES ARE NOT REAL OR AT LEAST NEAR-REAL, GIVE THEM
APPROXIMATE VALUE BUT SET INDIC TO 0
C
IF(DISC.LT.0.) GO TO 65
DISC=SQRT(DISC)
E1=DIF+H(N,N)+DISC
E2=E1-DISC-DISC
C
CHECK FOR CONVERGENCE OF THE 2 BY 2 SUBMATRIX
C
IF(NQ.NE.NM1) GO TO 70
C
THE 2 BY 2 SUBMATRIX HAS CONVERGED OR N=2
C
EVAL(NM1)=E1
INDIC(NM1)=1
EVAL(N)=E2
INDIC(N)=1
GO TO 66

```

Jul 13 18:46 1984 dbldif Page 30

```

C
C EIGENVALUES ARE NOT REAL. ASSIGN THEM THEIR REAL PARTS
C BUT SET INDIC TO 0 (- EIGENVALUES NOT FOUND)
C
65 EVAL(NM1)=DIF+H(N,N)
INDIC(NM1)=0
EVAL(N)=EVAL(NM1)
INDIC(N)=0
66 IF(N.EQ.2) RETURN
C
DEFLATE THE MATRIX BY DECREMENTING N
C
N=N-2
GO TO 10
C
THE 2 BY 2 SUBMATRIX HAS NOT CONVERGED
C
COMPUTE THE ORIGIN SHIFT BY THE IMPROVED SHIFT STRATEGY
(SEE WILKINSON PG.511)
C
DETERMINE WHICH OF E1 AND E2 IS CLOSER TO H(N,N)
C
70 TEMP=E1-H(N,N)
D1=TEMP*TEMP
TE1:P=E2-H(N,N)
D2=TEMP*TEMP
IF(D2.LT.D1) E1=E2
C
CHECK IF E1 HAS SETTLED DOWN TO AT LEAST ONE BINARY PLACE
C
TEMP=1.
IF(E1*E1.GT.1.E-200) TEMP=(E1-E0)/E1
IF(TEMP*TEMP.GT..25) GO TO 80
C

```

```

C E1 HAS SETTLED DOWN TO AT LEAST ONE BINARY PLACE
C PERFORM THE ORIGIN SHIFT OF THE IMPROVED SHIFT STRATEGY
C
C ZETA=E1
C GO TO 90
C
C E1 HAS NOT SETTLED DOWN TO AT LEAST ONE BINARY PLACE
C INCREMENT THE SHIFT COUNTER
C
80 NOGO=NOGO+1
ZETA=0.
C
C PERFORM THE DETERMINANT SHIFT WHEN NOGO=MIN(10,N-NQ+1)
C IF(NOGO.LT.N-NQ+1.AND.NOGO.LT.10) GO TO 90
C
C DETSHI COMPUTES THE SHIFT WHEN IT IS DETERMINED THAT
C SEVERAL EIGENVALUES HAVE THE SAME MODULUS
C
CALL DETSHI(H,NQ,N,NDIM,ZETA)
NOGO=0
C
C SINGLE COMPLEX QR ITERATION
C
C
C PERFORM THE ORIGIN SHIFT(I.E. H=H-ZETA*I)
C
90 DO 100 I=NQ,N
100 H(I,I)=H(I,I)-ZETA

```

JUL 13 18:46 1984 dbldif Page 31

```

C
C REDUCE H TO UPPER TRIANGULAR FORM BY USE OF PLANE ROTATIONS
C
IF(NM1.LT.NQ) GO TO 130
DO 120 I=NQ,NM1
DIAG=H(I,I)
SUBDIA=H(I+1,I)
H(I+1,I)=0.
KAP=SQRT(DIAG*DIAG+SUBDIA*SUBDIA)
MU(I)=DIAG/KAP
NU(I)=SUBDIA/KAP
H(I,I)=KAP
IP1=I+1
DO 110 J=IP1,N
TEMP=H(I,J)
H(I,J)=MU(I)*TEMP+NU(I)*H(IP1,J)
110 H(IP1,J)=MU(I)*H(IP1,J)-NU(I)*TEMP
120 CONTINUE
C
C ALL DIAGONAL ELEMENTS ARE NOW REAL EXCEPT THE LAST
C
C CHANGE THE LAST ELEMENT TO ITS MODULUS
C
130 DIAG=H(N,N)
KAP=ABS(DIAG)
IF(KAP.NE.0.) GO TO 140
MU(N)=1.
GO TO 150
140 H(N,N)=KAP
MU(N)=DIAG/KAP
150 NU(N)=0.
C
C FORM THE MATRIX RQ OF THE QR ALGORITHM

```

```

C      IF(NM1.LT.NQ) GO TO 170
C      DO 160 J=NQ,NM1
C          JP1=J+1
C          DO 160 I=NQ,JP1
C              TEMP=H(I,J)
C              H(I,J)=MU(J)*TEMP+NU(J)*H(I,JP1)
C 160      H(I,JP1)=MU(J)*H(I,JP1)-NU(J)*TEMP
C
C      COMPLETE THE OPERATION BY MULTIPLYING THE LAST COLUMN BY
C      MU(N) AND REMOVE THE ORIGIN SHIFT(I.E. H=H+ZETA*I)
C
C 170      DO 180 I=NQ,N
C          H(I,N)=MU(N)*H(I,N)
C 180      H(I,I)=H(I,I)+ZETA
C
C      INCREMENT THE ITERATION COUNTER
C
C      KOUNT=KOUNT+1
C      E0=E1
C      IF(KOUNT.LE.50) GO TO 20
C
C      NO CONVERGENCE IN THE PRINCIPAL SUBMATRIX IN ROWS NQ THROUGH N
C
C      DO 190 I=NQ,N
C          EVAL(I)=0.
C 190      INDIC(I)=0
C      IF(NQ.EQ.1) RETURN
C      N=NQ-1
C      GO TO 10

```

Jul 13 18:46 1984 dbldif Page 32

```

END
C*****SUBROUTINE DETSHI(H,NQ,N,NDIM,ZETA)
C
C      CALCULATION OF THE DETERMINANT OF THE PRINCIPAL SUBMATRIX IN ROWS
C      NQ THROUGH N OF THE UPPER HESSENBERG MATRIX H BY HYMANS METHOD
C
C      REFERENCE-THE ALGEBRAIC EIGENVALUE PROBLEM,WILKINSON,PGS.426-429
C
C      DIMENSION H(NDIM,1)
C      COMMON /QR/ X(200)
C      NM1=N-1
C      TRACE=0.
C      X(NQ)=1.
C
C      SCALAR CALCULATION LOOP
C
C      DO 20 J=NQ,N
C
C      ACCUMULATE THE TRACE
C
C      TRACE=TRACE+H(J,J)
C
C      SUM IS COMPUTED USING PREVIOUSLY COMPUTED SCALARS ONLY
C
C      SUM=0.
C      DO 10 I=NQ,J
C 10      SUM=SUM+H(I,J)*X(I)
C
C      WHEN J=N, SUM IS THE K OF HYMANS METHOD
C
C      IF(J.EQ.N) GO TO 30

```

```

C COMPUTE THE NEXT SCALAR
C
C JP1=J+1
20 X(JP1)=-SUM/H(JP1,J)
C COMPUTE THE DETERMINANT
C
30 DET=SUM
DO 40 J=NQ,NM1
40 DEI=DET*H(J+1,J)
C
C THE PRINCIPAL SUBMATRIX IN ROWS NQ THROUGH N IS
C ASSUMED TO HAVE EIGENVALUES OF EQUAL MODULUS
C
C COMPUTE THE MODULUS
C
ABSDET=ABS(DET)
R=ABSDET**(1./(N-NQ+1))
C
C SHIFT TO THE PERIMETER OF THE CIRCLE ON WHICH THE EIGENVALUES LIE
C
ZETA=R
IF (TRACE.NE.0.) ZETA=TRACE*R/ABS(TRACE)
RETURN
END
*****SUBROUTINE VECTR(W,N,NDIM,SUBDIA,EV,EVECT,INDIC)
C
C THE EIGENVECTOR OF THE UPPER HESSENBERG MATRIX H CORRESPONDING
C TO THE EIGENVALUE EV IS FOUND BY INVERSE ITERATION

```

Jul 13 18:46 1984 dbldif Page 33

```

C W IS DESTROYED IN CALCULATIONS
C
C REFERENCE-THE CALCULATION OF THE EIGENVECTORS OF A GENERAL
C COMPLEX MATRIX BY INVERSE ITERATION, J. M. VARAH
C MATHEMATICS OF COMPUTATION, VOL. 22, NO. 104, OCT. 1968
C
C DIMENSION W(NDIM,1)
C DIMENSION SUBDIA(1),EVECT(1)
C COMMON /QR/ INDEX(200),B(200)
C DATA TWOPI/6.283185307179586/
C EPS = 2.0**(-48)
C
C SMALL IS A MACHINE DEPENDENT CONSTANT DEFINED AS FOLLOWS-
C SMALL=SQUARE ROOT OF THE SMALLEST NORMALIZED POSITIVE MACHINE NUMB
C
C SMALL = 1.0E-1232
C
C CHECK THE ORDER OF THE MATRIX
C
C IF(N.NE.1) GO TO 10
C
C THE MATRIX IS OF ORDER 1
C
C EVECT(1)=1.
C INDIC=2
C RETURN
C
C INITIALIZATION OF VARIABLES
C
10 NM1=N-1
HNORM=0.
DO 30 I=1,N

```

```

ROWSUM=0.
IF(I.GT.1) ROWSUM=ABS(SUBDIA(I-1))
DO 20 J=I,N
20 ROWSUM=ROWSUM+ABS(W(I,J))
HNORM=AMAX1(ROWSUM,HNORM)
30 INDEX(I)=I

C
C      ZERO IS USED IN PLACE OF A ZERO PIVOT IN THE GAUSS ELIMINATION
C      AND IN THE CALCULATION OF A STOPPING CRITERION (TOL) FOR THE
C      INVERSE ITERATION

C      ZERO=EPS*HNORM
IF(ZERO.EQ.0.) ZERO=SMALL
TOL=1./(100.*N*ZERO)

C      FORM THE MATRIX TO BE REDUCED (I.E. W=H-EV*I)

C      DO 50 I=1,N
50 W(I,I)=W(I,I)-EV
DO 60 I=1,NM1
60 W(I+1,I)=SUBDIA(I)

C      REDUCE THE MATRIX TO UPPER TRIANGULAR FORM BY GAUSS ELIMINATION

C      DO 110 K=1,NM1
KP1=K+1
BIG=0.
DO 70 I=K,KP1
IACT=INDEX(I)
RS=W(IACT,K)*W(IACT,K)
IF(RS.LE.BIG) GO TO 70
BIG=RS

```

Jul 13 18:46 1984 dbldif Page 34

```

IDXPIV=I
70 CONTINUE
IF(BIG.NE.0.) GO TO 80
KACT=INDEX(K)
W(KACT,K)=ZERO
GO TO 110
80 IF(IDXPIV.EQ.K) GO TO 90
J=INDEX(K)
INDEX(K)=INDEX(IDXPIV)
INDEX(IDXPIV)=J
90 KACT=INDEX(K)
PIVOT=W(KACT,K)
IACT=INDEX(KP1)
QUOT=W(IACT,K)/PIVOT
DO 100 J=KP1,N
100 W(IACT,J)=W(IACT,J)-QUOT*W(KACT,J)
110 CONTINUE
KACT=INDEX(N)
IF(W(KACT,N).EQ.0.) W(KACT,N)=ZERO

C      INVERSE ITERATION

C      THETA=TWOPI/N
DO 160 K=1,N
KM1=K-1
DO 120 L=1,N
LM1=L-1
120 B(L)=COS(KM1*LM1*THETA)+SIN(KM1*LM1*THETA)
IACT=INDEX(N)
EVECT(N)=B(N)/W(IACT,N)

```

```

DO 140 IBACK=1,NM1
I=NM1-IBACK+1
IACT=INDEX(I)
SUM=0.
IP1=I+1
DO 130 J=IP1,N
130 SUM=SUM+W(IACT,J)*EVECT(J)
140 EVECT(I)=(B(I)-SUM)/W(IACT,I)

C      COMPUTE THE NORM OF THE VECTOR
C
BIG=0.
DO 150 I=1,N
RS=EVECT(I)*EVECT(I)
IF(BIG.GE.RS) GO TO 150
BIG=RS
IBIG=I
150 CONTINUE

C      CHECK FOR CONVERGENCE
C
IF(SQRT(BIG).GT.TOL) GO TO 170
160 CONTINUE

C      NO CONVERGENCE
C
RETURN

C      CONVERGENCE HAS BEEN ACHIEVED
C      NORMALIZE THE VECTOR, SET THE INDICATOR, AND RETURN
C
170 SUM=EVECT(IBIG)
DO 180 I=1,N

```

Jul 13 18:46 1984 dbldif Page 35

```

180 EVECT(I)=EVECT(I)/SUM
INDIC=2
RETURN
ENd
*****
SUBROUTINE LU2(A,N,NDIM,IR,IC)
C
C      PROGRAM TO PERFORM FULLY PIVOTED LU DECOMPOSITION OF GENERAL
C      REAL ARRAY A
C
DIMENSION A(NDIM,1)
DIMENSION IR(1), IC(1)
DO 10 I = 1,N
IR(I) = I
10 IC(I) = I
K = 1
L = K
M = K
XMAX=ABS(A(K,K))
DO 100 I = K,N
DO 100 J = K,N
Y=ABS(A(I,J))
IF( XMAX .GE. Y) GO TO 100
XMAX = Y
L = I
M = J
100 CONTINUE
DO 1000 K = 1,N
IRL = IR(L)

```

```

IR(L) = IR(K)
IR(K) = IRL
ICM = IC(M)
IC(M) = IC(K)
IC(K) = ICM
IF( L .EQ. K ) GO TO 300
DO 200 J = 1,N
B = A(K,J)
A(K,J) = A(L,J)
200 A(L,J) = B
300 IF( M .EQ. K ) GO TO 500
DO 400 I = 1,N
B = A(I,K)
A(I,K) = A(I,M)
400 A(I,M) = B
500 C = 1./A(K,K)
A(K,K) = C
IF( K .EQ. N ) GO TO 1000
K1 = K + 1
XMAX=ABS(A(K1,K1))
L = K1
M = K1
DO 600 I = K1,N
A(I,K) = C*A(I,K)
DO 800 I = K1,N
B = A(I,K)
DO 800 J = K1,N
A(I,J) = A(I,J) - B*A(K,J)
Y=ABS(A(I,J))
IF( XMAX .GE. Y ) GO TO 800
XMAX = Y
L = I
M = J
800 CONTINUE

```

Jul 13 18:46 1984 dbldif Page 36

```

1000 CONTINUE
RETURN
END
C*****
C SUBROUTINE DATASET (A,N2P,N1,GMAPP)
C COMMON /DATA/ ALPHA1,ALPHA2, BETA1,BETA2, BOXX, H, RNU,
1RKAPA1,RKAPA2, DT
C COMMON /PI/ PI
C COMMON /LADDR/ LZN,LTN,LPN,LZO,LTO,LPY,LZX,LZY,LEIG,LE,LH,LEV,
1LSN,LSO
DIMENSION A(N2P,1), GMAPP(1)
C
C PROGRAM TO INITIALIZE THE RUN (EITHER IN PHYSICAL OR FOURIER SPACE)
C PHYSICAL SPACE: KDATA=1 --> A (I,J) ... POINT (I-1, J-1)
C FOURIER SPACE : KDATA=2 --> CA(I,J) ... MODE (I-1, J-1)
C
C ALSO, IF YOU WANT TO SPECIFY THE MEAN FLOW (KSM=2), DO IT HERE.
C
COMMON /MFLOW/ U(129)
C
ALPHA1 = 10.0
ALPHA2=5.
C
C UNSTABLE CASE:
BETA1 = 1.0
BETA2=1.
RLAMDA = 1.35
BOXX = 2.0*PI*RLAMDA

```

```

FX = 2.0*PI/BOXX
C ASSUMING THAT GSET IS CALLED. IF LMAPP = 1, H SHOULD BE SET TO
C A DEFINED POSITIVE NUMBER.
C
H = GMAPP(N2P)
C COMPUTE VISCOSITY AND HEAT CONDUCTIVITY FROM PRANDTL NUMBER
C AND THE RATIO OF RAYLEIGH NUMBER TO ITS CRITICAL VALUE:
C
PRANDTL = 10.0
SCHMIDT=1.
RRATIO = 20.0
W = ALPHA1*BETA1*PRANDTL*4.0/(27.0*RRATIO)
RNU = SQRT (ABS(W))
RKAPA1 = RNU/PRANDTL
RKAPA2=RNU/SCHMIDT
C DETERMINE DIFFUSION LIMIT FOR THE TIME STEP AND COMPARE IT TO
C THE CHOSEN VALUE:
C
DT = 0.0001
X = FX*FLOAT(N1/2-1)
Y = FLOAT(N2P-1)
DTMAX = 1.0/((X*X+Y*Y) * AMAX1 (RNU,RKAPA))
PRINT 170,DTMAX,DT
170 FORMAT (* DIFFUSION CONSTRAINT ON DT: DTMAX = *,F10.6,/
1           * ACTUAL DT CHOSEN FOR THIS RUN: DT = *,F10.6)
NW = N2P*N1
C
XX = BOXX/FLOAT(N1)
C
EPS = 1.0
DO 20 J=1,N1
  X = FLOAT(J-1)*2.0*PI/FLOAT(N1)

```

Jul 13 18:46 1984 dbldif Page 37

```

DO 10 I=1,N2P
  Y = FLOAT(I-1)*PI/FLOAT(N2P-1)
  A(I,J) = EPS*SIN(X)*SIN(Y)
10  CONTINUE
20  CONTINUE
C
PRINT 100
100 FORMAT (///* INITIAL STREAMFUNCTION IS SIN(X) SIN(Y)*//)
CALL OUT (A,LPN,NW)
DO 40 J=1,N1
  DO 30 I=1,N2P
    A(I,J) = 0.0
30  CONTINUE
40  CONTINUE
PRINT 200
200 FORMAT (* INITIAL TEMPERATURE PERTURBATION ZERO*)
CALL OUT (A,LTN,NW)
DO 401 J=1,N1
  DO 301 I=1,N2P
    A(I,J) = 0.0
301  CONTINUE
401 CONTINUE
PRINT 2010
2010 FORMAT (* INITIAL SALT PERTURBATION ZERO*)
CALL OUT (A,LSN,NW)
RETURN
END

```

```

***** SUBROUTINE RUNINFO *****
COMMON /OPTIONS/ N1,N2,NUMBER,KDATA,LMAPP,NOTEMP,KTA,KTAPE,KSM,
1      KPP,KZP,KTP,KUP,KVP,KPF,KZF,KTF,KUF,KVF,KPRESS,NOALS,NOSALT
COMMON /DATA/ ALPHA1,ALPHA2,BETA1,BETA2,BOXX,H,RNU,RKAPA1,RKAPA2,
1DT,DTT,T,NSTEPS
C THIS SUBROUTINE TELLS US WHAT OCEAN WILL BE DOING:
C
C      ZERO = 1.0E-200
PRINT 100
C
C      MORESTEP = NUMBER-NSTEPS+1
PRINT 300,N1,N2,MORESTEP,KTA
IF (KSM.EQ.1) PRINT 301
IF (KSM.EQ.2) PRINT 302
C
C      GO TO (1,2,3), KDATA
PRINT 400
GO TO 10
1      PRINT 401
GO TO 10
2      PRINT 402
GO TO 10
3      PRINT 403
C
10     GO TO (11,123,123), LMAPP
PRINT 500
GO TO 20
11    PRINT 501
GO TO 20
123   PRINT 523
C
20     GO TO (21), NOTEMP
PRINT 600

```

Jul 13 18:46 1984 dbldif Page 38

```

21     GO TO 30
PRINT 601
C
30     GO TO (31,32,33), KTAPE
PRINT 700
GO TO 40
31    PRINT 701
GO TO 40
32    PRINT 702
GO TO 40
33    PRINT 703
C
40     IF (KPP.NE.1) PRINT 801,KPP
IF (KZP.NE.1) PRINT 802,KZP
IF (KTP.NE.1) PRINT 803,KTP
IF (KUP.NE.1) PRINT 804,KUP
IF (KVP.NE.1) PRINT 805,KVP
IF (KPF.NE.1) PRINT 806,KPF
IF (KZF.NE.1) PRINT 807,KZF
IF (KTF.NE.1) PRINT 808,KTF
IF (KUF.NE.1) PRINT 809,KUF
IF (KVF.NE.1) PRINT 810,KVF
IF (KPRESS.NE.1) PRINT 811,KPRESS
C
C      IF (NOALS.EQ.1) PRINT 901

```

```

REY = 1.0/(RNU+ZERO)
RH = H*H*H*H*ABS(ALPHA1*BETA1)/(RNU*RKAPA1+ZERO)
C CRITICAL RAYLEIGH NUMBER IS (PI**4) * 27/4:
C
RHCrit = 657.5113648
RRATIO = RH/RHCrit
PR = RNU/(RKAPA1+ZERO)
PRINT 1000, BOXX,H,RNU,RKAPA1,RKAPA2,ALPHA1,ALPHA2,BETA1,
1BETA2,REY,RH,RRATIO,PR,DT,T
C
RETURN
C
C FORMAT STATEMENTS:
C
100 FORMAT(12X,* 000000      CCCCCC   EEEEEE    AAAAAA   N   N*/,
2     12X,*0      0      C       C   E       A       A   NN   N*/
3     12X,*0      0      C       C   E       A       A   N   N   N*/
4     12X,*0      0      C       C   EEEE    AAAAAAAA   N   N   N*/
5     12X,*0      0      C       C   E       A       A   N   N   N*/
6     12X,*0      0      C       C   E       A       A   N   N   NN*/
7     12X,* 000000      CCCCCC   EEEEEE    A       A   N   N   N*)
C
C
300 FORMAT (/* THIS RUN USES *,I3,* BY *,I3,* GRID, AND DOES*,I5,
1   * TIME STEPS*,*, TIME AVERAGING IS DONE EVERY*,I3,* STEPS*)
301 FORMAT (* MEAN FLOW IS INCLUDED IN CALCULATIONS*)
302 FORMAT (* MEAN FLOW IS SPECIFIED BY USER*)
C
400 FORMAT (* KDATA HAS ILLEGAL VALUE*)
401 FORMAT (* INITIAL DATA IN PHYSICAL SPACE*)
402 FORMAT (* INITIAL DATA IN FOURIER SPACE*)
403 FORMAT (/* THIS IS A RESTART RUN*/)
C
500 FORMAT (* LMAPP HAS AN ILLEGAL VALUE*)
501 FORMAT (* LINEAR MAPPING IN BOTH DIRECTIONS*)
523 FORMAT (* NON-LINEAR MAPPING IN Y DIRECTION*)
Jul 13 18:46 1984 dbldif Page 39

C
600 FORMAT (* TEMPERATURE FIELD INCLUDED IN CALCULATIONS*)
601 FORMAT (* CALCULATIONS DONE WITHOUT TEMPERATURE FIELD*)
C
700 FORMAT (/* NO HISTORY TAPE WILL BE WRITTEN*/
701 FORMAT (/* HISTORY TAPE WILL INCLUDE OPERATOR MATRICES ONLY*/)
702 FORMAT (/* HISTORY TAPE WILL NOT INCLUDE OPERATOR MATRICES*/)
703 FORMAT (/* HISTORY TAPE WILL INCLUDE ALL IMPORTANT ARRAYS*/)
C
801 FORMAT (* OUTPUT PHYSICAL STREAMFUNCTION EVERY*,I4,* TIME STEPS*)
802 FORMAT (* OUTPUT PHYSICAL VORTICITY  EVERY*,I4,* TIME STEPS*)
803 FORMAT (* OUTPUT PHYSICAL TEMPERATURE  EVERY*,I4,* TIME STEPS*)
804 FORMAT (* OUTPUT PHYSICAL X VELOCITY  EVERY*,I4,* TIME STEPS*)
805 FORMAT (* OUTPUT PHYSICAL Y VELOCITY  EVERY*,I4,* TIME STEPS*)
806 FORMAT (* OUTPUT FOURIER STREAMFUNCTION EVERY*,I4,* TIME STEPS*)
807 FORMAT (* OUTPUT FOURIER VORTICITY  EVERY*,I4,* TIME STEPS*)
808 FORMAT (* OUTPUT FOURIER TEMPERATURE  EVERY*,I4,* TIME STEPS*)
809 FORMAT (* OUTPUT FOURIER X VELOCITY  EVERY*,I4,* TIME STEPS*)
810 FORMAT (* OUTPUT FOURIER V VELOCITY  EVERY*,I4,* TIME STEPS*)
811 FORMAT (* OUTPUT PHYSICAL PRESSURE   EVERY*,I4,* TIME STEPS*)
C
901 FORMAT (* DE-ALIASING TURNED ON.*)
C
1000 FORMAT (/*/* PHYSICAL PARAMETERS FOR THIS RUN ARE:*,/,
1   * BOX DIMENSIONS : *,F10.6,* BY *,F10.6,/,
```

```

2   * VISCOSITY (NU) : *,E24.14./,
3   * HEAT CONDUCTIVITY (KAPA) : *,2E24.14./,
4   * ALPHA AND BETA : *,2F10.6,* AND*,2F10.6./,
5   * REYNOLDS NUMBER : *,E24.14./,
6   * RAYLEIGH NUMBER : *,E24.14./,
7   * RATIO RAY/RAY(CRITICAL) : *,E24.14./,
8   * PRANDTL NUMBER : *,E24.14.//,* THE TIME STEP IS*,
9   F10.6./,* WE BEGIN AT T ==*,F10.6.///,1H ,60(2H+-),//)
END

```

```

C*****SUBROUTINE SHUF (A,N2P,N1,WORK)
DIMENSION A(N2P,1),WORK(1)
DO 10 J = 1,N1,2
  DO 20 I = 1,N2P
    II = I+I
    WORK(II-1) = A(I,J)
20    WORK(II) = A(I,J+1)
    DO 10 I = 1,N2P
      A(I,J) = WORK(I)
10    A(I,J+1) = WORK(I+N2P)
RETURN
END

```

```

C*****SUBROUTINE UNSHUF (A,N2P,N1,WORK)
DIMENSION A(N2P,1),WORK(1)
DO 10 J = 1,N1,2
  DO 20 I = 1,N2P
    WORK(I) = A(I,J)
20    WORK(I+N2P) = A(I,J+1)
    DO 10 I = 1,N2P
      II = I+I
      A(I,J) = WORK(II-1)
10    A(I,J+1) = WORK(II)
RETURN
END

```

```

C*****SUBROUTINE MEAN (CA,U,G)
DIMENSION U(1),G(1)

```

Jul 13 18:46 1984 dbldif Page 40

```

COMPLEX CA(1)
COMMON /NMBRS/ NSKIP(4), N2P
DO 10 I = 1,N2P
  U(I) = -G(I)*REAL(CA(I))
10 RETURN
END

```

```

C*****SUBROUTINE RPRINT (R,NAME,N1,N2)
DIMENSION R(N1,1)
PRINT 100, NAME
100 FORMAT (/,1H ,131(1H*),//,* -----> OUTPUT *,A8)
C

```

C (THIS SUBROUTINE CAN BE USED TO REPORT ANYTHING ABOUT FIELDS GIVEN
C IN THE REAL ARRAY R IF YOU MODIFY IT TO DO SO.)

```

C
  RETURN
END

```

```

C*****SUBROUTINE CPRINT (C,NAME,N1,N2)
COMPLEX C(N1,1)

```

```

C
  PRINT 100, NAME,N1,N2
  PRINT 200, (1, I = 1,N1)
  PRINT 300

```

```

DO 40 J = 1,N2
PRINT 400, J,(C(I,J), I = 1,N1)
40 CONTINUE
RETURN
C
100 FORMAT (*0PRINTOUT OF *,A10,*(*,I3,*,*I3,*) AS A COMPLEX ARRAY:*)
200 FORMAT (9X,*I =*,3(* (REAL)*,I10,7X,*((IMAG)*,7X),/(,26X,I3,2I37))
300 FORMAT (* J =*)
400 FORMAT (I6,3(3X,2E17.7),/(,9X,2E17.7,3X,2E17.7,3X,2E17.7))
C
END
C*****
```

```

SUBROUTINE PSIPHY (CA,N2P,N1H,EX,G,U,WORK,UU)
COMMON /EXSIZE/ NMAX2
COMMON /DATA/ SKIP(2), FX
COMPLEX UU(1), CA(N2P,1), EX(1), CF
DIMENSION U(1), G(1), WORK(1)
COMMON /FT/ NPTS, NSKIP, MTRN, MSKIP, ISIGN, LOG, IEXSHFT
CALL RPRINT (CA,8HPSI PHYS,N2P,1)
FX2 = 2.*FX
DO 5 I = 1,N2P
      CA(I,1) = 0.
DO 10 J = 2,N1H
      CF = 1./CMPLX(0.,FX2*FLOAT(J-1))
      DO 10 I = 1,N2P
            CA(I,J) = CA(I,J)*CF
10   NPTS = N1H+N1H
      NSKIP = N2P
      MTRN = N2P
      MSKIP = 1
      ISIGN = 1
      LOG = LOG2(NPTS)
      IEXSHFT = NMAX2/NPTS
      CALL CSR (CA,EX)
      DO 20 I = 1,N2P
            UU(I) = U(I)/G(I)
20   N2 = N2P-1
      NPTS = N2+N2
      NSKIP = 1
```

```

MTRN = 1
MSKIP = 1
ISIGN = -1
LOG = LOG2(NPTS)
IEXSHFT = NMAX2/NPTS
CALL SCSC (UU,EX,WORK)
FX2 = -1./(16.*FLOAT(N2))
DO 30 I = 2,N2
      FF = FX2/FLOAT(I-1)
      UU(I) = CMPLX(AIMAG(UU(I))*FF,-REAL(UU(I))*FF)
30   UU(1) = 0.
      UU(N2P) = 0.
      ISIGN = 1
      CALL ACAC (UU,EX,WORK)
      DO 40 I = 1,N2P
            RF = REAL(UU(I))+AIMAG(UU(I))
            CF = CMPLX(RF, RF)
            DO 40 J = 1,N1H
                  CA(I,J) = CA(I,J)+CF
40   N2PD = 2*N2P
      N1 = N1H+N1H
      CALL UNSHUF (CA,N2P,N1,WORK)
      CALL TELTIME
```

Jul 13 18:46 1984 dbldif Page 41

```

CALL PLOTME (CA,WORK,NCALL)
RETURN
END
C*****
SUBROUTINE ZETAPHY (A,N2PD,N1H,EX,WORK)
DIMENSION A(N2PD,1), EX(1), WORK(1)
COMMON /ZPLT/ NCALL
DATA NCALL /0/
CALL CSR (A,EX)
N2P = N2PD/2
CALL RPRINT (A,8HZETA PHY,N2P,1)
N1 = N1H+N1H
CALL UNSHUF (A,N2P,N1,WORK)
CALL TELTIME
DO 3 J=1,N1H
3   WORK(J) = 0.0
DO 4 I=1,N2PD
  DO 4 J=1,N1H
4     WORK(J) = WORK(J)+A(I,J)*A(I,J)
SUMSQ=0.0
DO 5 J=1,N1H
5   SUMSQ = SUMSQ+WORK(J)
ZRMS = SQRT (SUMSQ/(FLOAT(N1)*FLOAT(N2P)))
PRINT 6, ZRMS
6 FORMAT (/* SQRT (AVERAGE ZETASQUARED) =*,F30.15)
NCALL = NCALL+1
CALL PLOTME (A,WORK,NCALL)
RETURN
END
C*****
SUBROUTINE TEMPHY (A,N2PD,N1H,EX,WORK)
COMMON /LADDR/ LDUMMY, LTN
DIMENSION A(N2PD,1), WORK(1)
COMMON /DATA/ ALPHA1,ALPHA2,BETA
COMMON /MAPPING/ GMAPP(129)
COMMON /TPLT/ NCALL
DATA NCALL /0/
NW = N2PD*N1H
N2P = N2PD/2
N1 = N1H+N1H

```

Jul 13 18:46 1984 dbldif Page 42

```

CALL RPRINT (A,8HTEMP PHY,N2P,1)
CALL IN (A,LTN,NW)
CALL CSR (A,EX)
DO 3 J=1,N1H
3   WORK(J) = 0.0
DO 4 I=1,N2PD
  IPH = (I+1)/2
  ADJUST = BETA*GMAPP(IPH)
C WHEN INTERESTED IN TEMPERATURE DISTURBANCE ONLY, USE THIS INSTEAD:
C   ADJUST = 0.0
  DO 4 J=1,N1H
    WORK(J) = WORK(J)+A(I,J)*A(I,J)
4   A(I,J) = 0.5*A(I,J)-ADJUST
SUMSQ=0.0
DO 5 J=1,N1H
5   SUMSQ = SUMSQ+WORK(J)
ZRMS = 0.5*SQRT (SUMSQ/(FLOAT(N1)*FLOAT(N2P)))
PRINT 6, ZRMS
6 FORMAT (/* SQRT (AVERAGE TEMPERATURESQUARED) =*,F30.15)
NCALL = NCALL+1
CALL UNSHUF (A,N2P,N1,WORK)
CALL TELTIME

```

```

CALL PLOTME (A,WORK,NCALL)
RETURN
END
C*****
SUBROUTINE SALTPHY (A,N2PD,N1H,EX,WORK)
COMMON /LADDR/ LZN,LTN,LPN,LZO,LTO,LPY,LZX,LZY,LEIG,LE,LH,LEV,
1LSN,LSO
DIMENSION A(N2PD,1), WORK(1)
COMMON /DATA/ ALPHA1,ALPHA2,BETA1,BETA2
COMMON /MAPPING/ GMAPP(129)
COMMON /TPLT/ NCALL
DATA NCALL /0/
NW = N2PD*N1H
N2P = N2PD/2
N1 = N1H+N1H
CALL RPRINT (A,BHSALT PHY,N2P,1)
CALL IN (A,LSN,NW)
CALL CSR (A,EX)
DO 3 J=1,N1H
3   WORK(J) = 0.0
DO 4 I=1,N2PD
  IPH = (I+1)/2
  ADJUST = BETA2*GMAPP(IPH)
C WHEN INTERESTED IN SALT DISTURBANCE ONLY, USE THIS INSTEAD:
C   ADJUST = 0.0
  DO 4 J=1,N1H
    WORK(J) = WORK(J)+A(I,J)*A(I,J)
4   A(I,J) = 0.5*A(I,J)-ADJUST
SUMSQ=0.0
DO 5 J=1,N1H
5   SUMSQ = SUMSQ+WORK(J)
ZRMS = 0.5*SQRT (SUMSQ/(FLOAT(N1)*FLOAT(N2P)))
PRINT 6, ZRMS
6 FORMAT /* SQRT (AVERAGE SALTSQUARED) =*,F30.15)
NCALL = NCALL+1
CALL UNSHUF (A,N2P,N1,WORK)
CALL TELTIME
CALL PLOTME (A,WORK,NCALL)
RETURN
END
C*****

```

Jul 13 18:46 1984 dbldif Page 43

```

SUBROUTINE UVELPHY (A,N2PD,N1H,WORK)
COMMON /LADDR/ LDUMMY(5), LPY
DIMENSION A(N2PD,1), WORK(1)
COMMON /DATA/ DUMMY1(2), BOXX, DUMMY2(7)
DO 5 I=1,N2PD
5   WORK(I) = 0.0
NW = N2PD*N1H
N2P = N2PD/2
N1 = N1H+N1H
CALL RPRINT (A,BHUVELPHY ,N2P,1)
CALL TELTIME
CALL IN (A,LPY,NW)
DO 10 J = 1,N1H
  DO 10 I = 1,N2PD
    WORK(I) = AMAX1 (WORK(I), ABS(A(I,J)) )
10  DO 20 I=2,N2PD
20  WORK(1) = AMAX1 (WORK(1), WORK(I))
CALL UNSHUF (A,N2P,N1,WORK)
C AT THIS POINT A(N2P,N1) CONTAINS 2.0*U
C

```

```

UMAX = 0.5*WORK(1)+1.0E-200
DTMAX = 0.2*(BOXX/FLOAT(N1))/UMAX
PRINT 100, DTMAX
100 FORMAT (/* U VELOCITY: DT MUST BE LESS THAN 0.2 MAX((DX/DJ)/U) =*
1 ,F20.15)
RETURN
END
C*****
SUBROUTINE VVELPHY (A,N2PD,N1H,WORK)
COMMON /LADDR/ LDUMMY(2), LPN
DIMENSION A(N2PD,1), WORK(1)
COMMON /MAPPING/ GMAPP(I29),G(129)
NW = N2PD*N1H
N2P = N2PD/2
N1 = N1H+N1H
CALL RPRINT (A,8HVVELPHY ,N2P,1)
CALL TELTIME
CALL IN (A,LPN,NW)
CALL UNSHUF (A,N2P,N1,WORK)
C
C AT THIS POINT A(N2P,N1) CONTAINS 2.0*V
C
CALL TESTDT (A,N2P,N1,WORK,G)
RETURN
END
C*****
SUBROUTINE TESTDT (A,N2P,N1,WORK,G)
COMMON /PI/ PI
DIMENSION A(N2P,N1),WORK(N2P),G(N2P)
C
C FIND MAXIMUM DT:
C
DO 10 I=1,N2P
10 WORK(I) = 0.0
DO 20 J=1,N1
   DO 20 I=1,N2P
      20 WORK(I) = AMAX1 (WORK(I), ABS(G(I)*A(I,J)))
C
C G WAS 1.0/(DY/DX), AND X GOES FROM 0.0 TO PI, HENCE IF WE WANT
C ((DY/DI)/V)=(DY/DX)(DX/DI)/V)=PI/(N1*G(I)*V(I))
C REMEMBER, A CONTAINS 2.0*V:
DO 30 I=2,N2P

```

Jul 13 18:46 1984 db1dif Page 44

```

30      WORK(1) = AMAX1 (WORK(1), WORK(I))
C
C SO, WORK(1) = MAX (G(I)*2*V(I,J)):
C DT(MAX) = 0.2* MIN (DY/V) = 0.2*2.0*PI/(N1*WORK(1)):
C DTMAX = 0.4*PI/(FLOAT(N1)*WORK(1)+1.0E-1000)
C PRINT 200, DTMAX
200 FORMAT (/* V VELOCITY: DT MUST BE LESS THAN 0.2 MAX((DY/DI)/V) =*
1 ,F20.15)
RETURN
END
C*****
SUBROUTINE PSIFOU (CA,N2P,N1H,CWORK)
COMMON /LADDR/ LDUMMY(2), LPN
COMPLEX CA(N2P,1),CWORK(1)
NW = N2P*N1H*2
CALL IN (CA,LPN,NW)
CALL CPRINT (CA,7HPSI FOU,N2P,N1H)
RETURN
END
C*****

```

```
SUBROUTINE ZETAFOU (CA,N2P,N1H,CWORK)
COMMON /LADDR/ LZN
COMPLEX CA(N2P,1),CWORK(1)
NW = N2P*N1H*2
CALL IN (CA,LZN,NW)
CALL CPRINT (CA,8HZETA FOU,N2P,N1H)
RETURN
END
```

```
C*****
```

```
SUBROUTINE SALTFOU (CA,N2P,N1H,CWORK)
COMMON /LADDR/ LZN,LTN,LPN,LZO,LTO,LPY,LZX,LZY,LEIG,LE,LH,LEV,
ILSN,LSO
COMPLEX CA(N2P,1), CWORK(1)
NW = N2P*N1H*2
CALL IN (CA,LSN,NW)
CALL CPRINT (CA,8HSALT FOU,N2P,N1H)
RETURN
END
```

```
C*****
```

```
SUBROUTINE TEMPFOU (CA,N2P,N1H,CWORK)
COMMON /LADDR/ LDUMMY, LTN
COMPLEX CA(N2P,1), CWORK(1)
NW = N2P*N1H*2
CALL IN (CA,LTN,NW)
CALL CPRINT (CA,8HTEMP FOU,N2P,N1H)
RETURN
END
```

```
C*****
```

```
SUBROUTINE UVELFOU (CA,N2P,N1H,G,EX,CWORK)
COMMON /FT/ NDUMMY(4), ISIGN
COMPLEX CA(N2P,1),CWORK(1),EX(1)
```

```
DIMENSION G(1)
DO 10 I = 1,N2P
    DO 10 J = 1,N1H
```

```
10    CA(I,J) = G(I)*CA(I,J)
```

```
ISIGN = -1
CALL SCSC (CA,EX,CWORK)
```

```
ISIGN = 1
```

```
F = .125/FLOAT(N2P-1)
```

```
DO 20 J = 1,N1H
```

```
    DO 20 I = 1,N2P
```

```
20    CA(I,J) = F*CA(I,J)
        CALL CPRINT (CA,8HUVEL FOU,N2P,N1H)
```

Jul 13 18:46 1984 dbldif Page 45

```
RETURN
END
```

```
C*****
```

```
SUBROUTINE VVELFOU (CA,N2P,N1H,FX,CWORK)
```

```
COMMON /LADDR/ LDUMMY(2), LPN
```

```
COMPLEX CA(N2P,1),CWORK(1)
```

```
NW = N2P*N1H*2
```

```
CALL IN (CA,LPN,NW)
```

```
F = .0.
```

```
DO 20 J = 1,N1H
```

```
    DO 10 I = 1,N2P
```

```
10    CA(I,J) = CMPLX(-F*AIMAG(CA(I,J)), F*REAL(CA(I,J)))
```

```
20    F = F+FX
```

```
CALL CPRINT (CA,8HVVEL FOU,N2P,N1H)
```

```
RETURN
```

```
END
```

```
C*****
```

```
SUBROUTINE EIGIN
```

```
C
```

C THESE SUBROUTINES WILL TAKE CARE OF THE HISTORY TAPE.
 C SINCE THEY ARE NOT WRITTEN YET, ONLY AN ERROR MESSAGE IS PRINTED.

```

C
C ENTRY EIGOUT
C ENTRY DATA1N
C ENTRY DATAOUT
C PRINT 100
100 FORMAT(* SUBROUTINES WHICH HANDLE THE TAPE DO NOT YET EXIST.*.,.
1      * THIS IS AN ERROR EXIT, PLEASE CHECK OPTIONS SPECIFIED*.,.
2      * IN SUBROUTINE START.*)
STOP TAPE
END

```

```

*****SUBROUTINE PRESSET (CA,G,GMAPP,CWORK,EX,EVAL,IR,IC,E,N2M,LPRESS)
COMMON /SOLV/ LVECT, LSKIP, NVECT, NVSKIP
COMMON /FT/ NPTS, NSKIP, MTRN, MSKIP, ISIGN, LOG, IEXSHFT
COMMON /EXSIZE/ NMAX2
COMMON /PI/ PI
COMMON /DATA/ SKIP(3), H
COMMON /LADDR/ LDUMMY(8), LEIG, LE, LH, LEV,
1LSN,LSO
DIMENSION INDIC(127)
DIMENSION G(1), GMAPP(1), EVAL(1)
DIMENSION IR(1), IC(1), E(N2M,1)
COMPLEX CA(1), EX(1), CWORK(1)

```

```

C PREPARE THE MATRIX TO COMPUTE EIGENVALUES AND EIGENVECTORS
C USING THE MAPPING FUNCTION. MAPPING FUNCTION IS STORED IN GMAPP
C AND IT GOES FROM Ø. TO H
C VECTOR G CONTAINS ITS DERIVATIVE UPON THE RETURN FROM EIGSET
C

```

```

N2 = N2M+1
N2P = N2+1
H = GMAPP(N2P)
F = H/FLOAT(N2)
DO 10 I = 1,N2P
   G(I) = GMAPP(I)-F*FLOAT(I-1)
10  CA(I) = CMPLX (G(I), Ø.)
CA(1) = Ø.
CA(N2P) = Ø.
NPTS = 2*(N2P-1)
NSKIP = 1

```

Jul 13 18:46 1984 db1dif Page 46

```

MTRN = 1
MSKIP = 1
IEXSHFT = NMAX2/NPTS
LOG = LOG2 (NPTS)
ISIGN = -1
CALL ACAC (CA,EX,CWORK)
DO 20 I = 1,N2P
   CA(I) = CA(I)*CMPLX(Ø., FLOAT(I-1))
20  CA(N2P) = Ø.
ISIGN = 1
CALL SCSC (CA,EX,CWORK)
F = 8.*FLOAT(N2P-1)*H/PI
FN = 8.*FLOAT(N2P-1)
DO 30 I = 1,N2P
   G(I) = FN/(F+REAL(CA(I)))
30  NWM = N2M*N2M
    NWEIG = N2M*(N2M+3)
    AFACT = 2./FLOAT(N2)

```

```

PIFACT = PI/FLOAT(N2)
NPTS = 2*N2
NSKIP = 1
MTRN = 1
MSKIP = 1
LOG = LOG2 (NPTS)
IEXSHFT = NMAX2/NPTS
FF = 1./(16.*FLOAT(N2*N2))
KE = LE
DO 1000 J = 1,N2M
    FACT = PIFACT*FLOAT(J)
    DO 40 I = 1,N2P
        CWORK(I) = CMPLX(0.,-G(I)*FLOAT(J)*SIN(FACT*FLOAT(I-1)))
40      ISIGN = -1
        CALL ACAC (CWORK,EX,E)
        DO 50 I = 1,N2
50      CWORK(I) = CWORK(I)*CMPLX(0.,FLOAT(I-1))
        CWORK(N2P) = 0.
        ISIGN = 1
        CALL SCSC (CWORK,EX,E)
        DO 60 I = 1,N2P
60      CWORK(I) = G(I)*CWORK(I)
        ISIGN = -1
        CALL SCSC (CWORK,EX,E)
        DO 70 I = 2,N2
70      E(I-1,1) = REAL(CWORK(I))*FF
        CALL OUT (E,KE,N2M)
1000   KE = KE + N2M
        CALL IN (E,LE,NWM)
C
C COMPUTE EIGENVALUES AND EIGENVECTORS, CHECK SUCCESS AND STOP IF NOT 0.
C
        CALL LCMRQR (E,N2M,N2M,EVAL,1,LH,LEV,INDIC)
        LVECT = N2M
        LSKIP = 2
        NVECT = 1
        NVSKIP = 1
        DO 110 I = 1,N2M
            IF (INDIC(I) - 1) 1000, 1001, 110
1000   PRINT 210, I,INDIC(I)
        STOP EIGS
C
C IF AN EIGENVECTOR WAS NOT FOUND, COMPUTE IT THE OLD WAY
C
1001   CALL OUT (E,LEV,NWM)

```

Jul 13 18:46 1984 dbldif Page 47

```

        CALL IN (E,LE,NWM)
        DO 1010 J = 1,N2M
            CWORK(J) = 1.
1010   E(J,J) = E(J,J)-EVAL(I)
        CALL LU2 (E,N2M,N2M,IR,IC)
        DO 1020 K = 1,3
            CALL SOLVE2 (CWORK,E,N2M,IR,IC)
            SUM = 0.
        DO 1030 J = 1,N2M
            SUM = SUM + REAL(CWORK(J))*REAL(CWORK(J))
1030   IF (SUM .EQ. 0.) GO TO 1020
            F = 1./SQRT(SUM)
            DO 1040 J = 1,N2M
                CWORK(J) = REAL(CWORK(J))*F
1040   CONTINUE
            XLARGE = 0.
            DO 1060 J = 1,N2M

```

```

        IF (ABS(REAL(CWORK(J))) .LT. XLARGE) GO TO 1060
        XLARGE = ABS(REAL(CWORK(J)))
        IXL = J
        CONTINUE
1060     FACT = 1./REAL(CWORK(IXL))
        CALL IN (E,LEV,NWM)
        DO 1050 J = 1,N2M
           E(J,1) = REAL(CWORK(J))*FACT
        PRINT 220, I,INDIC(I),EVAL(I)
110     CONTINUE
210     FORMAT (* EIGSET ERROR, INDIC(*,I3,*),=*,I3)
220     FORMAT (///,* LCMRQR FAILURE, INDIC(*,I3,*),=*,I2,/,
1           * EIGENVECTOR CORRESPONDING TO EIGENVALUE *,E15.7,
2           * HAS BEEN COMPUTED THE OLD WAY*)
C
C LU TRANSFORM EIGENVECTORS AND STORE RESULTS IN LCM
C ASSUMING THAT EVAL, IR, IC AND E OCCUPY NWEIG CONSECUTIVE LOCATIONS
C
        CALL LU2 (E,N2M,N2M,IR,IC)
        CALL OUT (EVAL,LPRESS,NWEIG)
        RETURN
        END
C*****SUBROUTINE TELTIME
C      COMMON /DATA/ DUMMY(8),TIME
C      PRINT 100,TIME
100     FORMAT (* TIME = *,F20.6)
        RETURN
        END
C*****SUBROUTINE PRESURE (LU, LV, LU2, LPM, NWPX, A, CA, WORK, CWORK, G,
1       EX, EVAL, IR, IC, E, FX, N1C, N1R, N2R, LMAPP)
C
C LU IS LCM ADDRESS OF U VELOCITY (EVEN IN Y DIR.; PRESSURE RETURNED HE
C LV ..... V ..... (ODD ....)
C LU2 ..... WORKSPACE
C LPM ..... THE PRESSURE MATRIX
C
C A IS THE REAL VERSION OF THE ARRAY
C CA ..... COMPLEX ..... (A AND CA MUST BE THE SAME)
C
C WORK IS REAL WORKSPACE
C CWORK .. COMPLEX .....; THEY MAY BE THE SAME
C
C G CONTAINS DZ/DY, I.E. THE INVERSE OF THE DERIVATIVE OF THE MAPPING FN
EX IS THE ARRAY OF COMPLEX EXPONENTIALS USED BY FFTS

```

Jul 13 18:46 1984 dbldif Page 48

```

C
C FX IS A DIFF. FACTOR (THE SAME AS IN MAIN PRG.)
C N1CD IS THE FIRST DIMENSION OF CA (= 2**N +1)
C N1RD ..... A (HAS TO BE 2*N1CD)
C N2R IS THE SECOND DIMENSION FOR A (SAME AS N2C)
C
C NOTE: THE NOTATION (N1 V N2) IS NOT THE SAME AS IN THE MAIN PRG., BUT
C THERE IS NO SPECIFIC REASON FOR THAT.
C
        COMMON /FT/ NPTS, NSKIP, MTRN, MSKIP, ISIGN, LOG, IEXS
        COMMON /EXSIZE/ NMAX
        COMMON /SOLV/ LVECT, LSKIP, NVECT, NVSKIP, GSOLVE(1)
C
        DIMENSION A(N1R,1), WORK(1), G(1), EVAL(1), IR(1), IC(1), E(1)
        COMPLEX CA(N1C,1), CWORK(1), EX(1)
C

```

```

C THESE ARE SOME RANDOM FACTORS WE USE...
C
NW = N1R*N2R
NWH = NW/2
NWLIN = N1R
N2C = N2R
N2RH = N2R/2
N2CH = N2RH
N1CM = N1C-1
N1CMM = N1C-2
JH = N2C/2
JHP = JH+1
FXSQ = FX*FX
FU2 = 1.0/(128.0*FLOAT(N1CM)*FLOAT(N2C))
FUV = 8.0*FU2*FU2
FV2 = 2.0*FU2*FUV
LVECT = N1CMM
LSKIP = 2
NVECT = 2
NVSKIP = 1
C INPUT U AND SQUARE IT, GO TO THE FOURIER SPACE AFTERWARDS:
C
CALL IN (A,LU,NW)
DO 20 I = 1,N1R
    DO 10 J = 1,N2R
        A(I,J) = A(I,J)*A(I,J)
10    CONTINUE
20    CONTINUE
C
NPTS = 2*N2C
NSKIP = N1C
MTRN = N1C
MSKIP = 1
ISIGN = -1
LOG = LOG2(NPTS)
IEXS = NMAX/NPTS
C
CALL RCS (A,EX)
C
NPTS = 2*N1CM
NSKIP = 1
MTRN = N2C
MSKIP = N1C
ISIGN = -1
LOG = LOG2(NPTS)

```

Jul 13 18:46 1984 dblcif Page 49

```

IEXS = NMAX/NPTS
C
CALL SCSC (A,EX,WORK)
C
C CA IS EQUIVALENCED TO A, SO THAT WE MAY DIFFERENTIATE:
C
DO 40 J = 1,N2C
    DXXF = FLOAT(J-1)*FLOAT(J-1)*FXSQ
    DO 30 I = 1,N1C
        CA(I,J) = DXXF*CA(I,J)
30    CONTINUE
40    CONTINUE
C
C STORE THE FOURIER TRANSFORM OF -D2U2/DX2:
C

```

```

CALL OUT (CA,LU2,NW)
C BRING IN U AND V FOR PRODUCT
C
KU = LU
KV = LV
DO 70 K = 1,2
    CALL IN (A(1,1),KU,NWH)
    CALL IN (A(1,JHP),KV,NWH)
    DO 60 I = 1,NIR
        DO 50 J = 1,N2RH
            A(I,J) = -A(I,J)*A(I,J+JH)
            A(I,J+JH) = A(I,J+JH)*A(I,J+JH)
50      CONTINUE
60      CONTINUE
C MOVE OUT UV AND V2:
C
CALL OUT (A(1,1),KU,NWH)
CALL OUT (A(1,JHP),KV,NWH)
KU = KU+NWH
KV = KV+NWH
70 CONTINUE
C CALL IN V2 TO COMPUTE DV2/DY
C
CALL IN (A,LV,NW)
C
NPTS = 2*N2C
NSKIP = N1C
MTRN = N1C
MSKIP = 1
ISIGN = -1
LOG = LOG2(NPTS)
IEXS = NMAX/NPTS
C
CALL RCS (A,EX)
C
NPTS = 2*N1CM
NSKIP = 1
MTRN = N2C
MSKIP = N1C
ISIGN = -1
LOG = LOG2(NPTS)
IEXS = NMAX/NPTS
C
CALL SCSC (CA,EX,WORK)
C
```

Jul 13 18:46 1984 dbldif Page 50

```

C COMPUTE THE Y PARTIAL DERIVATIVE:
C
DO 90 I = 1,N1CM
    DZF = FLOAT(I-1)
    DO 80 J = 1,N2C
        CA(I,J) = CMPLX(-DZF*AIMAG(CA(I,J)), DZF*REAL(CA(I,J)))
80      CONTINUE
90      CONTINUE
    DO 100 J = 1,N2C
        CA(N1C,J) = 0.0
100     CONTINUE
C
C GO BACK TO THE PHYSICAL SPACE...
C
```

```
NPTS = 2*N1CM
NSKIP = 1
MTRN = N2C
MSKIP = N1C
ISIGN = 1
LOG = LOG2(NPTS)
IEXS = NMAX/NPTS
C
C CALL ACAC (CA,EX,WORK)
C
NPTS = 2*N2C
NSKIP = N1C
MTRN = N1C
MSKIP = 1
ISIGN = 1
LOG = LOG2(NPTS)
IEXS = NMAX/NPTS
C
C CALL CSR (CA,EX)
C
C MULTIPLY BY DZ/DY...
C
DO 120 I = 1,N1R
    IPH = (I+1)/2
    GG = G(IPH)
    DO 110 J = 1,N2R
        A(I,J) = A(I,J)*GG
110    CONTINUE
120    CONTINUE
C
C BACK TO FOURIER...
C
ISIGN = -1
C
C CALL RCS (A,EX)
C
NPTS = 2*N1CM
NSKIP = 1
MTRN = N2C
MSKIP = N1C
ISIGN = -1
LOG = LOG2(NPTS)
IEXS = NMAX/NPTS
C
C CALL ACAC (CA,EX,WORK)
C
C CALL OUT (CA,LV,NW)
C
C START WITH THE PHYSICAL UV TO COMPUTE DUV/DX
```

Jul 13 18:46 1984 dbldif Page 51

```
C
C CALL IN (A,LU,NW)
C
NPTS = 2*N2C
NSKIP = N1C
MTRN = N1C
MSKIP = 1
ISIGN = -1
LOG = LOG2(NPTS)
IEXS = NMAX/NPTS
C
C CALL RCS (A,EX)
```

```

NPTS = 2*N1CM
NSKIP = 1
MTRN = N2C
MSKIP = N1C
ISIGN = -1
LOG = LOG2(NPTS)
IEXS = NMAX/NPTS

C CALL ACAC (A,EX,WORK)

C DO 140 J = 1,N2C
  DXF = FLOAT(J-1)*FX
  DO 130 I = 1,N1C
    CA(I,J) = CMPLX(-DXF*AIMAG(CA(I,J)), DXF*REAL(CA(I,J)))
130  CONTINUE
140  CONTINUE

C ADD THE FOURIER TRANSFORMS OF DUV/DX AND DV2/DY

C KU = LU+NWH
CA_L OUT (CA(1,JHP),KU,NWH)
KU = LU
KV = LV
DO 170 K = 1,2
  CALL IN (CA(1,JHP),KV,NWH)
  DO 160 I = 1,N1C
    DO 150 J = 1,N2CH
      CA(I,J) = FUV*CA(I,J)+FV2*CA(I,J+JH)
150  CONTINUE
160  CONTINUE
  CALL OUT (CA(1,1),KU,NWH)
  KU = KU+NWH
  KV = KV+NWH
  IF (K .EQ. 1) CALL IN (CA(1,1),KU,NWH)
170  CONTINUE

C FIND D/DY(DUV/DX+DV2/DY):
C CALL IN (CA,LU,NW)

C DO 190 I = 1,N1CM
  DZF = FLOAT(I-1)
  DO 180 J = 1,N2C
    CA(I,J) = CMPLX(-DZF*AIMAG(CA(I,J)), DZF*REAL(CA(I,J)))
180  CONTINUE
190  CONTINUE
  DO 200 J = 1,N2C
    CA(N1C,J) = 0.0
200  CONTINUE

```

Jul 13 18:46 1984 db1dif Page 52

```

C GO BACK TO THE PHYSICAL SPACE...
C

NPTS = 2*N1CM
NSKIP = 1
MTRN = N2C
MSKIP = N1C
ISIGN = 1
LOG = LOG2(NPTS)
IEXS = NMAX/NPTS

C CALL SCSC (CA,EX,WORK)

```

```

NPTS = 2*N2C
NSKIP = N1C
MTRN = N1C
MSKIP = 1
ISIGN = 1
LOG = LOG2(NPTS)
IEXS = NMAX/NPTS
C
C CALL CSR (CA,EX)
C MULTIPLY BY DZ/DY...
C
DO 220 I = 1,NIR
 1PH = (I+1)/2
 GG = G(IPH)
 DO 210 J = 1,N2R
   A(I,J) = A(I,J)*GG
210  CONTINUE
220  CONTINUE
C
C BACK TO FOURIER...
C
ISIGN = -1
C
CALL RCS (A,EX)
C
NPTS = 2*N1CM
NSKIP = 1
MTRN = N2C
MSKIP = N1C
ISIGN = -1
LOG = LOG2(NPTS)
IEXS = NMAX/NPTS
C
CALL SCSC (CA,EX,WORK)
C
C FIND -D2U2/DX2 -2*D2UV/DYDX -D2V2/DY2 = DEL2 P:
C
KU = LU+NWH
CALL OUT (CA(1,JHP),KU,NWH)
KU = LU
KU2 = LU2
DO 250 K = 1,2
  CALL IN (CA(1,JHP),KU2,NWH)
  DO 240 I = 1,N1C
    DO 230 J = 1,N2CH
      CA(I,J) = FU2*CA(I,J+JH) - CA(I,J)
230  CONTINUE
240  CONTINUE
CALL OUT (CA(1,1),KU,NWH)
KU = KU+NWH

```

Jul 13 18:46 1984 dbldif Page 53

```

KU2 = KU2+NWH
IF (K .EQ. 1) CALL IN (CA(1,1),KU,NWH)
250 CONTINUE
C
C THIS GAVE US DEL2 P; NOW WE SOLVE FOR P:
C
CALL IN (EVAL,LPM,NWPX)
KU = LU
DO 270 J = 1,N2C
  CALL IN (CWORK,KU,NWLINE)
  IF (LMAPP .NE. 1) CALL SOLVE2 (CWORK(2),E,N1CMM,IR,IC)

```

```

F = FLOAT(J-1)*FLOAT(J-1)*FXSQ
DO 260 I = 2,NICM
  CWORK(I) = CWORK(I) / (EVAL(I-1)-F)
260 CONTINUE
IF (LMAPP .NE. 1) CALL LUMULT2 (CWORK(2),E,NICMM,IR,IC)
WZERO = 0.0
IF (F .NE. 0.0) WZERO = -CWORK(1)/F
CWORK(1) = WZERO
CWORK(NIC) = 0.0
CALL OUT (CWORK,KU,NWLINE)
KU = KU+NWLINE
270 CONTINUE
C THAT'S ALL FOLKS! THE PRESSURE IS KNOWN AND WE PROUDLY PRESENT IT
C   CALL IN (A,LU,NW)
C DO NOT FORGET!!! WE ARE STILL IN THE FOURIER SPACE, SO SCSC/CSR SHOULD
C BE CALLED.
C
NPTS = 2*N1CM
NSKIP = 1
MTRN = N2C
MSKIP = N1C
ISIGN = 1
LOG = LOG2(NPTS)
IEXS = NMAX/NPTS
C   CALL SCSC (A,EX,WORK)
C
NPTS = 2*N2C
NSKIP = N1C
MTRN = N1C
MSKIP = 1
ISIGN = 1
LOG = LOG2(NPTS)
IEXS = NMAX/NPTS
C   CALL CSR (A,EX)
C WE GOT BACK THE PHYSICAL VALUES SO WE CAN UNRAVEL THE MESS AND PLOT P:
C
N1PHY = N1R/2
N2PHY = N2R*2
CALL UNSHUF (A, N1PHY, N2PHY, WORK)
CALL RPRINT (A,8HPRES.PHY,N1PHY,1)
CALL TELTIME
CALL PLOTME (A,WORK,NCALL)
RETURN
END
*****
SUBROUTINE PLOTME (Z,WORK,NCALL)

```

Jul 13 18:46 1984 dbldif Page 54

```

COMMON /DATA/ ALPHA1,ALPHA2,BETA1,BETA2,BOXX,H
COMMON /OPTIONS/ NX,NY
DIMENSION Z(1), WORK(1)
COMMON /PLTDEF/ IDPAT, DXA, DXB, DYA, DYB
C THIS SUBROUTINE CAN BE USED FOR PLOTTING OF FIELDS IN PHYSICAL
C SPACE. CONTOUR PLOTS ARE USUALLY BEST.
C
ASPECT = BOXX/H
A = (DXB-DXA) / (DYB-DYA)

```

```

XMIN = DXA
XMAX = DXB
YMIN = DYB
YMAX = DYB
IF (A.LT.ASPECT) GO TO 10
SPANH = 0.5*(DYB-DYA) * ASPECT
RMEAN = 0.5*(DXA+DXB)
XMIN = RMEAN-SPANH
XMAX = RMEAN+SPANH
GO TO 30
10 SPANH = 0.5*(DXB-DXA) / ASPECT
RMEAN = 0.5*(DYA+DYB)
YMIN = RMEAN-SPANH
YMAX = RMEAN+SPANH
30 CALL SET (XMIN,XMAX,YMIN,YMAX,1.0,FLOAT(NX+1),1.0,FLOAT(NY+1),1)
CALL CONREC (Z,NY+1,NY+1,NX,0.0,0.0,0.0,1.0,0)
CALL FRAME
RETURN
END
*****
C***** FUNCTION LCMREQ (LARGE)
PARAMETER (LCMSIZE = 250000)
COMMON /MEMORY/ MEMSIZE,RM(LCMSIZE)
IF (LARGE.GT.LCMSIZE) GO TO 10
MEMSIZE = LARGE
LCMREQ = 1
PRINT 100,LARGE
100 FORMAT (/* LCM REQUESTED:*,I7,* WORDS*,//)
RETURN
C
10 PRINT 11, MEMSIZE, LARGE
11 FORMAT (/* LCMREQ: REQUEST EXCEEDS MEMORY SIZE.*/
1      ,* MEMORY SIZE:*,I7,* ; WORDS REQUESTED:*,I22)
STOP
END
*****
C***** SUBROUTINE IN (ARRAY,LOC,NW)
COMMON /MEMORY/ MEMSIZE,RM(250000)
DIMENSION ARRAY(1)
IF ((LOC.LT.1).OR.((LOC+NW-1).GT.MEMSIZE).OR.(NW.LT.1)) GO TO 100
LOCM=LOC-1
DO 10 I=1,NW
    ARRAY(I) = RM(LOCM+I)
10 CONTINUE
RETURN
C
100 PRINT 101,LOC,NW
101 FORMAT (/* IN: ILLEGAL REQUEST. LOCATION:*,I22,
1      * ; NUMBER OF WORDS:*,I22,* ; MEMSIZE:*,I7)
STOP
END
*****

```

Jul 13 18:46 1984 dbldif Page 55

```

C***** SUBROUTINE OUT (ARRAY,LOC,NW)
COMMON /MEMORY/ MEMSIZE, RM(250000)
DIMENSION ARRAY(1)
IF ((LOC.LT.1).OR.((LOC+NW-1).GT.MEMSIZE).OR.(NW.LT.1)) GO TO 100
LOCM = LOC-1
DO 10 I=1,NW
    RM(LOCM+I) = ARRAY(I)
10 CONTINUE

```

```

RETURN
C
100 PRINT 101,LOC,NW,MEMSIZE
101 FORMAT (///* OUT: ILLEGAL REQUEST. LOCATION:*,I22,
           * ; NUMBER OF WORDS:*,I22,* ; MEMSIZE:*,I7)
STOP
END
SUBROUTINE LUMULT2(F,A,NDIM,IR,IC)
DIMENSION A(NDIM,1),F(1),IR(1),IC(1)
COMMON /SOLV/ LVECT,LSKIP,NVECT,NVSKIP,G(127)
N1=LVECT+1
LASTV=NVECT*NVSKIP
DO 1000 KK=1,LASTV,NVSKIP
DO 100 I=1,LVECT
ICI=(IC(I)-1)*LSKIP+KK
100 G(I)=F(ICI)
DO 200 I=1,LVECT
I1=I+1
B=G(I)/A(I,I)
IF(I.EQ.LVECT) GO TO 200
DO 201 J=I1,LVECT
201 B=B+A(I,J)*G(J)
200 G(I)=B
DO 300 IT=1,LVECT
I=N1-IT
I1=I-1
B=G(I)
IF(I.EQ.1) GO TO 300
DO 301 J=1,I1
301 B=B+A(I,J)*G(J)
300 G(I)=B
DO 400 I=1,LVECT
IRI=(IR(I)-1)*LSKIP+KK
400 F(IRI)=G(I)
1000 CONTINUE
RETURN
END

```

```

SUBROUTINE SOLVE2(F,A,NDIM,IR,IC)
DIMENSION A(NDIM,1),F(1),IR(1),IC(1)
COMMON /SOLV/ LVECT,LSKIP,NVECT,NVSKIP,G(127)
N1=LVECT+1
LASTV=NVECT*NVSKIP
DO 1000 KK=1,LASTV,NVSKIP
DO 100 I=1,LVECT
IRI=(IR(I)-1)*LSKIP+KK
100 G(I)=F(IRI)
DO 400 I=2,LVECT
I1=I-1
B=G(I)
DO 300 J=1,I1
300 B=B-A(I,J)*G(J)
400 G(I)=B
DO 700 IT=1,LVECT
I=N1-IT

```

Jul 13 18:46 1984 dbldif Page 56

```

I1=I+1
B=G(I)
IF(I.EQ.LVECT) GO TO 700
DO 600 J=I1,LVECT
600 B=B-A(I,J)*G(J)
700 G(I)=B*A(I,I)
DO 900 I=1,LVECT
ICI=(IC(I)-1)*LSKIP+KK

```

```

900 F(ICI)=G(I)
1000 CONTINUE
RETURN
END

```

3. MULTI-DOMAIN DIFFUSION PROBLEMS

In this Chapter, we describe several numerical codes based on spectral methods for the solution of multi-domain and multi-dimensional heat-diffusion problems, including Bridgman-type crystal growth configurations. Three computer codes are included:

- A. A test program to solve the nonlinear Liouville equation

$$u_{xx} + u_{yy} + \gamma e^u = 0 \quad (3.1)$$

on the unit square with Dirichlet boundary conditions. This equation is solved by the routine LIVIL listed below using a double Chebyshev series collocation (pseudospectral) method. The purpose of this code is to test both the sparse spectral inversion procedure outlined in Chapter 1 and to test the pseudo-arclength continuation method for generation of new solutions from old ones. The results listed at the end of the program listing show the efficacy of this approach.

- B. The program DIFEQ solves the general two-dimensional variable coefficient heat equation using a pseudo-implicit method based on the sparse spectral approximation method outlined in Chapter 1. High-order implicit time integration methods are used to achieve the highest possible accuracy in time. Chebyshev spectral methods are used in both space

dimensions to maintain high spatial accuracy. Again, the sample results appended to the end of the listing are indicative of the very high degree of accuracy and efficiency achieved by these methods.

Again, the program DIFEQ is written in such a way that it is relatively straightforward to extend the code to three dimensions. However, the accuracy improvements available in three dimensions for given computational work are substantially higher than in two dimensions.

Further results on these problems are described by Moro & Orszag (1984).

C. The program CRYSTALG solves the problem of Bridgman-type crystal growth in a planar two-dimensional geometry. Here the basic equations are those given by Naumann (1982) to describe directional solidification experiments. The basic domain is subdivided into four sub-domains: (1) a hot zone in which the liquid phase is surrounded by conducting surfaces; (2) an adiabatic liquid zone; (3) an adiabatic solid phase zone; and (4) a conducting cold zone in the solid phase. The sample results given at the end of the listing were obtained with only an 8x8 array of Chebyshev polynomials in each zone. Nevertheless, the accuracy of the results is quite high as shown there.

The prgram CRYSTALG embodies both the spectral iteration procedure and the spectral patching procedures outlined in Chapter 1. Therefore, it is representative of the complexity of real

problems encountered in the thermal modeling of crystal growth. The combination of implicit time differencing, Chebyshev spectral series representations, and accurate domain patching extend to general three-dimensional problems and promise highly accurate numerical solutions to these problems.

Again, further results and technical details are given by Moro & Orszag (1984).

A. PROGRAM LIVIL AND TEST RESULTS

PROGRAM LIVIL

$\theta+2$ LIOUVILLE EQUATION -
 $\text{DEL}^2 U(X,Y) + \text{GAMA EXP}(U(X,Y)) = 0$ ON THE UNIT SQUARE

CONTINUATION IN GAMA UNTIL GAMA=6.6
 PSEUDOARCLENGTH CONTINUATION (IN SIGMA) AFTERWARDS

CHEBYSHEV COLLOCATION

```

PARAMETER {IX=16,IZ=16,NWK=4096)
PARAMETER {NX2=IX+2,NZ2=IZ+1,NXX=(IX-1)*(IZ-1)}
PARAMETER {ILA=3*(IZ-1)+1)
COMMON /PARAM/PI,DX,DZ,PIX,PIZ,NDIM,LDA,IWK
COMMON /FT/NPTS,NSKIP,MTRAN,MSKIP,ISIGN,LOG,IEX
COMMON /EXS/NMAX,EX(4096)
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
COMMON /SIZE/BOXX,BOXZ
REAL X1(NX2,NZ2),X2(NX2,NZ2),P(NXX),AP(NXX),DG(NXX)
1,R(NXX),AMTX(NXX,5),W(NXX),JAC(ILA,NXX),DGI(NXX)
2,Y(NXX),WORK(NWK),U(NXX),U0(NXX),DU0(NXX)
INTEGER IPVT(NXX)
PI=4.*ATAN(1.)
ERR=1.E-5
KSC=10
KSPC=20
LMAX=30
IWK=NWK
NX=IX
NZ=IZ
NDIM=NXX
NZHP=NZ/2+1
NXHP=NX/2+1
NXPP=NX+2
NXP=NX+1
NZP=NZ+1
NXM=NX-1
NZM=NZ-1
LDA=ILA
MAXNP=MAX0(NXP,NZP)
NTOT=NXPP*NZP
NMAX=2*MAX0(NX,NZ)
BOXX=2.
BOXZ=2.
WRITE(6,30)NX,NZ
FORMAT(/,* GRID *,I5,*X*,I5,/)
GAMA=0.8
SIGMA=0.2
DELTA=2.2
40 WRITE(6,40)GAMA,SIGMA,DELTA
FORMAT(/,* GAMA *,F8.2,* SIGMA *,F10.3,* DELTA *,F5.1,/)
DO 60 I=1,NDIM
60 U(I)=0.
DX=1./FLOAT(NX)
DZ=1./FLOAT(NZ)
PIX=PI*DX
PIZ=PI*DZ
CALL EXSET(EX,NMAX)

CALL SETMAT(AMTX)
CALL CONT(WORK,AMTX,JAC,IPVT,X1

```

```

1,X2,R,P,AP,KSC,LMAX,ERR,GAMA,DELTA,W,U0,U)
1,AMTX,WORK,IPVT,NSKIP,MTRAN,MSKIP,ISIGN,LOG,IEX
1,X1,X2,R,P,AP,KSPC,LMAX,ERR,W,DU0,DG,DGI,U0,U)
1,STOP
END

```

```

SUBROUTINE JPR(WORK,X1,X2,W,Y,X)
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXN
COMMON /PARAM/P1,DX,DZ,PIX,PIZ,NDIM,LDA,IWK
REAL X1(NXPP,1),X2(NXPP,1),WORK(IWK)
1,X(NDIM),Y(NDIM),W(NDIM)
DO 110 J=1,NZM
J1=(J-1)*NXM
DO 100 I=1,NXM
X2(I+1,J+1)=Y(I+J1)
X1(I+1,J+1)=Y(I+J1)
CONTINUE
DO 180 J=1,NZP
X1(1,J)=0.
X2(1,J)=0.
X1(NXP,J)=0.
X2(NXP,J)=0.
DO 190 I=1,NXP
X1(I,1)=0.
X2(I,1)=0.
X1(I,NZP)=0.
X2(I,NZP)=0.
CALL X2DERIV(X1,WORK)
CALL Z2DERIV(X2,WORK)
DO 270 J=1,NZM
J1=(J-1)*NXM
DO 260 I=1,NXM
X(I+J1)=-4.*(X2(I+1,J+1)+X1(I+1,J+1))-W(I+J1)*Y(I+J1)
CONTINUE
RETURN
END

```

```

SUBROUTINE RES(WORK,X1,X2,W,Y,X)
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXN
COMMON /PARAM/P1,DX,DZ,PIX,PIZ,NDIM,LDA,IWK
REAL X1(NXPP,1),X2(NXPP,1),WORK(IWK)
1,X(NDIM),Y(NDIM),W(NDIM)
DO 110 J=1,NZM
J1=(J-1)*NXM
DO 100 I=1,NXM
X2(I+1,J+1)=Y(I+J1)
X1(I+1,J+1)=Y(I+J1)
CONTINUE
DO 180 J=1,NZP
X1(1,J)=0.
X2(1,J)=0.
X1(NXP,J)=0.
X2(NXP,J)=0.
DO 190 I=1,NXP
X1(I,1)=0.
X2(I,1)=0.
X1(I,NZP)=0.
X2(I,NZP)=0.
CALL X2DERIV(X1,WORK)
CALL Z2DERIV(X2,WORK)
DO 270 J=1,NZM
J1=(J-1)*NXM
DO 260 I=1,NXM
X(I+J1)=4.*(X2(I+1,J+1)+X1(I+1,J+1))+W(I+J1)
CONTINUE
RETURN
END

```

```

SUBROUTINE BNCG(WORK,AMTX,JAC,IPVT,X1,X2,R,P,AP,LMAX,ERR
1,L,ERR1,W,DGAMA,GAMA0,GAMA,SIGMA,DG,DGI,DU,U0,U)
COMMON /PARAM/PI,DX,DZ,PIX,PIZ,NDIM,LDA,IWK
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
REAL X1(NXPP,1),X2(NXPP,1),W(NDIM),DG(NDIM)
1,AP(NDIM),R(NDIM),P(NDIM),AMTX(NDIM,5),U0(NDIM)
2,JAC(LDA,NDIM),WORK(IWK),U(NDIM),DU(NDIM),DG(NDIM)
INTEGER IPVT(NDIM)
DO 20 I=1,NDIM
DG(I)=-EXP(U(I))
W(I)=-GAMA*Dg(I)
CALL RES(WORK,X1,X2,W,U,R)
RGAMA=0.
DO 80 I=1,NDIM
RGAMA=RGAMA-DU(I)*(U(I)-U0(I))
RGAMA=RGAMA-DGAMA*(GAMA-GAMA0)+SIGMA
CALL SCOPY(NDIM,R,1,P,1)
PGAMA=RGAMA
CALL SETJAC(AMTX,W,JAC)
CALL SGBFA(JAC,LDA,NDIM,NXM,NXM,IPVT,INFO)
CALL SCOPY(NDIM,DG,1,DGI,1)
CALL BELIM(JAC,DGI,DU,DGAMA,IPVT,P,PGAMA)
RR=SDOT(NDIM,R,1,P,1)+RGAMA+PGAMA
L=0
999 L=L+1
CALL JPR(WORK,X1,X2,W,P,AP)
DO 120 I=1,NDIM
120 AP(I)=AP(I)+PGAMA*Dg(I)
APGAMA=SDOT(NDIM,DU,1,P,1)+DGAMA*PGAMA
PMP=SDOT(NDIM,P,1,AP,1)+PGAMA*APGAMA
PMP=RR/PMP
R1R=RR
CALL SAXPY(NDIM,PMP,P,1,U,1)
GAMA=GAMA+PMP*PGAMA
DO 180 I=1,NDIM
DG(I)=-EXP(U(I))
W(I)=-GAMA*Dg(I)
CALL RES(WORK,X1,X2,W,U,R)
RGAMA=0.
DO 210 I=1,NDIM
210 RGAMA=RGAMA-DU(I)*(U(I)-U0(I))
RGAMA=RGAMA-DGAMA*(GAMA-GAMA0)+SIGMA
CALL SCOPY(NDIM,R,1,AP,1)
APGAMA=RGAMA
CALL SETJAC(AMTX,W,JAC)
CALL SCOPY(NDIM,DG,1,DGI,1)
CALL SGBFA(JAC,LDA,NDIM,NXM,NXM,IPVT,INFO)
CALL BELIM(JAC,DGI,DU,DGAMA,IPVT,AP,APGAMA)

RR=SDOT(NDIM,R,1,AP,1)+RGAMA*APGAMA
ERR1=SQRT(ABS(RR))
IF(L.GE.LMAX)RETURN
IF(ERR1.LE.ERR)RETURN
PMP=RR/R1R
DO 310 I=1,NDIM
310 P(I)=AP(I)+PMP*P(I)
PGAMA=APGAMA+PMP*PGAMA
GOTO 999

```

END

```

SUBROUTINE SETMAT(AMTX)
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
COMMON /PARAM/PI,DX,DZ,PIX,PIZ,NDIM,LDA,IWK
REAL AMTX(NDIM,5)
DX=4./(PI*PI*DX*DX)
DZ=4./(PI*PI*DZ*DZ)
DO 50 J=1,NZM
J1=(J-1)*NXM
Y1=1./SIN(PIZ*(FLOAT(J)-0.5))
Y2=DZZ/SIN(PIZ*FLOAT(J))
Y3=1./SIN(PIZ*(FLOAT(J)+0.5))
Y21=Y2*Y1
Y23=Y3*Y2
DO 40 I=1,NXM
X1=1./SIN(PIX*(FLOAT(I)-0.5))
X2=DXX/SIN(PIX*FLOAT(I))
X3=1./SIN(PIX*(FLOAT(I)+0.5))
X21=X2*X1
X23=X3*X2
AMTX(I+J1,5)=-Y23
AMTX(I+J1,1)=-Y21
AMTX(I+J1,4)=-X23
40 AMTX(I+J1,2)=-X21
CONTINUE
DO 80 I=1,NDIM
80 AMTX(I,3)=-AMTX(I,5)-AMTX(I,1)-AMTX(I,4)-AMTX(I,2)
DO 130 I=1,NDIM,NXM
AMTX(I+NXM-1,4)=0.
130 AMTX(I,2)=0.
DO 150 I=1,NXM
AMTX(I,1)=0.
150 AMTX(NDIM-NXM+I,5)=0.
RETURN
END

```

```

SUBROUTINE SETJAC(AMTX,U,JAC)
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
COMMON /PARAM/PI,DX,DZ,PIX,PIZ,NDIM,LDA,IWK
REAL JAC(LDA,NDIM),AMTX(NDIM,5),U(NDIM)
DO 20 J=1,NDIM

```

```

10 DO 10 I=1,LDA
10 JAC(I,J)=0.
CONTINUE
20 DO 40 I=1,NDIM
JAC(LDA,I)=AMTX(I+NXM,1)
JAC(2*NZM+1,I)=AMTX(I,3)-U(I)
JAC(2*NZM+2,I)=AMTX(I+1,2)
IF(I+1.GT.NDIM)GOTO 40
JAC(2*NZM,I+1)=AMTX(I,4)
IF(I+NZM.GT.NDIM)GOTO 40
JAC(NZM+1,I+NZM)=AMTX(I,5)
40 CONTINUE
RETURN
END

```

```

SUBROUTINE Z2DERIV(B,WORK)
COMMON /FT/NPTS,NSKIP,MTRAN,MSKIP,ISIGN,LOG,IEX
COMMON /EXS/NMAX,EX(1)
COMMON /CHEB/NCHEB,NSKCHEB,MCHEB,MSKCHEB,FZ
COMMON /SIZE/BOXX,BOXZ
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
REAL B(NXPP,1),WORK(8192)
NPTS=2*NZ
NXHP=NXPP/2
NSKIP=NXHP
MTRAN=NSKIP
MSKIP=1
ISIGN=-1
LOG=LOG2(NPTS)
IEX=NMAX/NPTS
NCHEB=NZ
NSKCHEB=NXHP
MCHEB=NXHP
MSKCHEB=1
FZ=1./SQRT(BOXZ*FLOAT(NZ))
CALL SCSC(B,EX,WORK)
CALL CHEBDIF(B,B)
CALL CHEBDIF(B,B)
ISIGN=1
CALL SCSC(B,EX,WORK)
RETURN
END

```

```

SUBROUTINE X2DERIV(B,WORK)
COMMON /FT/NPTS,NSKIP,MTRAN,MSKIP,ISIGN,LOG,IEX
COMMON /EXS/NMAX,EX(1)
COMMON /CHEBRL/NCHEB,NSKCHEB,MCHEB,MSKCHEB,FZ
COMMON /SIZE/BOXX,BOXZ
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
REAL B(NXPP,1),WORK(8192)
NPTS=2*NX
NSKIP=1
MTRAN=NZP
MSKIP=NXPP/2
ISIGN=-1
LOG=LOG2(NPTS)
IEX=NMAX/NPTS
NCHEB=NX
NSKCHEB=1
MCHEB=NZP
MSKCHEB=NXPP

```

```

FZ=1./SQRT(BOXX*FLOAT(NX))
CALL SRSR(B,EX,WORK)
CALL RLCHBDF(B,B)
CALL RLCHBDF(B,B)
ISIGN=1
CALL SRSR(B,EX,WORK)
RETURN
END

```

```

SUBROUTINE PACON(WORK,AMTX,JAC,IPVT,GAMA,SIGMA
1 X1,X2,R,P,AP,KSTEPS,LMAX,ERR,W,DUØ,DG,DGI,UØ,U)
COMMON /PARAM/PI,DX,DZ,PIX,PIZ,NDIM,LDA,IWK
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
REAL DG(NDIM),X1(NXPP,1),X2(NXPP,1),W(NDIM),U(NDIM)
1 AP(NDIM),R(NDIM),AMTX(NDIM,5),UØ(NDIM)
2 JAC(LDA,NDIM),WORK(IWK),DUØ(NDIM),DET(2),DGI(NDIM)
INTEGER IPVT(NDIM)
DGAMA=1.
K=0
999 K=K+1
WRITE(6,110)K
110 FORMAT(/,* STEP *,I5)
GAMAØ=GAMA
DGAMAØ=DGAMA
DO 320 I=1,NDIM
DGI(I)=-EXP(U(I))
DGI(I)=DGI(1)
320 W(I)=-GAMA*DGI(I)
CALL SCOPY(NDIM,U,1,UØ,1)
CALL SETJAC(AMTX,W,JAC)
CALL SGBFA(JAC,LDA,NDIM,NXM,NXM,IPVT,INFO)
CALL SGBDI(JAC,LDA,NDIM,NXM,NXM,IPVT,DET)
WRITE(6,280)DET(1),DET(2)
280 FORMAT(* DETERMINANT *,F10.5,* E *,F6.1)
CALL LCG(WORK,AMTX,JAC,IPVT,X1
1,X2,R,P,AP,LMAX,ERR,L1,ERR1,W,DGI,U)
WRITE(6,390)L1,ERR1
390 FORMAT(* NUMBER OF ITERATIONS FOR PREDICTOR AND ERROR *,I5,F18.10)
IF(ERR1.LT.1.E-3)GOTO 355
355 IF(L1.GE.LMAX)GOTO 555
DGAMA=1./SQRT(1.+SDOT(NDIM,U,1,U,1))
CALL SCOPY(NDIM,U,1,DGI,1)
CALL SSCAL(NDIM,-DGAMA,U,1)
DUØDU=SDOT(NDIM,DUØ,1,U,1)+DGAMAØ*DGAMA
IF(DUØDU.GE.Ø.Ø)GOTO 888
CALL SSCAL(NDIM,-1.,U,1)
DGAMA=-DGAMA
888 CALL SCOPY(NDIM,U,1,DUØ,1)
DO 160 I=1,NDIM
160 U(I)=UØ(I)+SIGMA*U(I)
GAMA=GAMAØ+SIGMA*DGAMA
WRITE(6,180)DGAMA,GAMA
180 FORMAT(* DERIVATIVE OF GAMA *,F16.8,* GAMA PREDICTED *,F12.6)
CALL BNCG(WORK,AMTX,JAC,IPVT,X1,X2,R,P,AP,LMAX,ERR,L1,ERR1
1,W,DGAMA,GAMAØ,GAMA,SIGMA,DG,DGI,DUØ,UØ,U)
NMID=(NDIM+1)/2
WRITE(6,210)GAMA,U(NMID)
210 FORMAT(* GAMA *,F8.4,* SOLUTION AT THE CENTRE *,E12.6)
WRITE(6,220)L1,ERR1
220 FORMAT(* NUMBER OF ITERATIONS AND ERROR *,I5,F18.10)
IF(ERR1.LT.1.E-3)GOTO 455
IF(L1.GE.LMAX)GOTO 555
455 IF(K.GE.KSTEPS)RETURN

```

```

380 DO 380 I=1,NDIM
DG(I)=-EXP(U(I))
GOTO 999

STOP
555 WRITE(6,780)GAMA
780 FORMAT(/,* ITERATION FAILED AT GAMA = *,F12.5,/)
STOP
END

```

```

SUBROUTINE BELIM(JAC,B,C,D,IPVT,X,Y)
COMMON /NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
COMMON /PARAM/PI,DX,DZ,PIX,PIZ,NDIM,LDA,IWK
REAL JAC(NDIM,LDA),B(NDIM),C(NDIM),X(NDIM)
INTEGER IPVT(NDIM)
CALL SGBSL(JAC,LDA,NDIM,NXM,NXM,IPVT,B,0)
CALL SGBSL(JAC,LDA,NDIM,NXM,NXM,IPVT,X,0)
CW=SDOT(NDIM,C,1:X,1)
CV=SDOT(NDIM,C,1:B,1)
Y=(Y-CW)/(D-CV)
DO 120 I=1,NDIM
120 X(I)=X(I)-Y*B(I)
RETURN
END

```

```

SUBROUTINE CONT(WORK,AMTX,JAC,IPVT,X1
1,X2,R,P,AP,KSTEPS,LMAX,ERR,GAMA0,DELTA,Y,W0,W)
COMMON /PARAM/PI,DX,DZ,PIX,PIZ,NDIM,LDA,IWK
COMMON /NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
REAL XI(NXPP,1),X2(NXPP,1),W(NDIM),WORK(IWK)
1,AP(NDIM),R(NDIM),P(NDIM),AMTX(NDIM,5),JAC(LDA,NDIM)
2 Y(NDIM) W0(NDIM)
INTEGER IPVT(NDIM)
K=0
999 K=K+1
CALL SCOPY(NDIM,W,1,W0,1)
GAMA=GAMA0+DELTA
WRITE(6,70)GAMA
70 FORMAT(/,* GAMA *,F10.3)
DO 120 I=1,NDIM
120 Y(I)=GAMA*EXP(W(I))
CALL SETJAC(AMTX,Y,JAC)
CALL SGBFA(JAC,LDA,NDIM,NXM,NXM,IPVT,INFO)
CALL CNCG(WORK,AMTX,JAC,IPVT,X1
1,X2,R,P,AP,LMAX,ERR,L1,ERR1,GAMA,Y,W)
WRITE(6,220)L1,ERR1
220 FORMAT(* NUMBER OF ITERATIONS AND ERROR *,I5,F18.10)
IF(K.GT.KSTEPS)RETURN
IF(L1.GE.LMAX)GOTO 222
GAMA0=GAMA
IF(GAMA.EQ.6.6)RETURN
GOTO 999
22 40 WRITE(6,340)GAMA
FORMAT(/,* CONTINUATION FAILED AT GAMA = *,F8.1,/)

CALL SCOPY(NDIM,W0,1,W,1)
RETURN
END

```

```

SUBROUTINE CNCG(WORK,AMTX,JAC,IPVT,X1
1,X2,R,P,AP,LMAX,ERR,L,ERR1,GAMA,W,X)
COMMON /PARAM/PI,DX,DZ,PIX,PIZ,NDIM,LDA,IWK
COMMON /NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
REAL X(NDIM),XI(NXPP,1),X2(NXPP,1),W(NDIM)
1,AP(NDIM),R(NDIM),P(NDIM),AMTX(NDIM,5)
REAL JAC(LDA,NDIM),WORK(IWK)
INTEGER IPVT(NDIM)
CALL RES(WORK,X1,X2,W,X,R)
CALL SCOPY(NDIM,R,1,P,1)
CALL SGBSL(JAC,LDA,NDIM,NXM,NXM,IPVT,P,0)
RR=SDOT(NDIM,R,1,P,1)
999 L=0
L=L+1
CALL JPR(WORK,X1,X2,W,P,AP)
PMP=SDOT(NDIM,P,1,AP,1)
PMP=RR/PMP
RIR=RR
CALL SAXPY(NDIM,PMP,P,1,X,1)
DO 180 I=1,NDIM
180 W(I)=GAMA*EXP(X(I))
CALL RES(WORK,X1,X2,W,X,R)
CALL SCOPY(NDIM,R,1,AP,1)
CALL SGBSL(JAC,LDA,NDIM,NXM,NXM,IPVT,AP,0)
RR=SDOT(NDIM,R,1,AP,1)
ERR1=SQRT(ABS(RR))
IF(L.LE.LMAX)RETURN
IF(ERR1.LE.ERR)RETURN
PMP=RR/RIR
DO 310 I=1,NDIM
310 P(I)=AP(I)+PMP*P(I)
GOTO 999
END

```

*Goodman
On Your Quality*

```

SUBROUTINE LCG(WORK,AMTX,JAC,IPVT,X1
1,X2,R,P,AP,LMAX,ERR,L,ERR1,W,Y,X)
COMMON /PARAM/P1,DX,DZ,PIX,PIZ,NDIM,LDA,IWK
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
REAL X(NDIM),X1(NXPP,1),X2(NXPP,1),W(NDIM)
1,AP(NDIM),R(NDIM),P(NDIM),AMTX(NDIM,5)
REAL JAC(LDA,NDIM),WORK(IWK),Y(NDIM)
INTEGER IPVT(NDIM)
CALL JPR(WORK,X1,X2,W,X,AP)
DO 60 I=1,NDIM
  R(I)=Y(I)-AP(I)
  CALL SGBSL(JAC,LDA,NDIM,NXM,NXM,IPVT,R,B)
  RR=SDOT(NDIM,R,1,R,1)
  L=B
999  L=L+1
  CALL JPR(WORK,X1,X2,W,R,AP)
  CALL SGBSL(JAC,LDA,NDIM,NXM,NXM,IPVT,AP,B)
  PMP=SDOT(NDIM,R,1,AP,1)
  PMP=RR/PMP
  R1R=RR
  CALL SAXPY(NDIM,PMP,R,1,X,1)
  CALL SAXPY(NDIM,-PMP,AP,1,R,1)
  CALL SCOPY(NDIM,R,1,AP,1)
  RR=SDOT(NDIM,R,1,AP,1)

```

```

ERR1=SQRT(ABS(RR))
IF(L.GE.LMAX)RETURN
IF(ERR1.LE.ERR)RETURN
GOTO 999
END

```

GRID 16X 16

GAMA 0.00 SIGMA 0.200 DELTA 2.2

GAMA	2.200	
NUMBER OF ITERATIONS AND ERROR	6	0.0000046618
GAMA	4.400	
NUMBER OF ITERATIONS AND ERROR	6	0.0000086363
GAMA	6.600	
NUMBER OF ITERATIONS AND ERROR	12	0.0000045700

STEP	1	
DETERMINANT	2.57134 E 701.0	
NUMBER OF ITERATIONS FOR PREDICTOR AND ERROR	5	0.0000047997
DERIVATIVE OF GAMA	0.25232219	GAMA PREDICTED 6.650464
GAMA	6.6473	SOLUTION AT THE CENTRE 0.110454E+01
NUMBER OF ITERATIONS AND ERROR	3	0.0000038720

STEP	2	
DETERMINANT	2.02756 E 701.0	
NUMBER OF ITERATIONS FOR PREDICTOR AND ERROR	5	0.0000059188
DERIVATIVE OF GAMA	0.22088597	GAMA PREDICTED 6.691505
GAMA	6.6884	SOLUTION AT THE CENTRE 0.114154E+01
NUMBER OF ITERATIONS AND ERROR	3	0.0000031507

STEP	3	
DETERMINANT	1.56724 E 701.0	
NUMBER OF ITERATIONS FOR PREDICTOR AND ERROR	5	0.0000078427
DERIVATIVE OF GAMA	0.18959305	GAMA PREDICTED 6.726297
GAMA	6.7232	SOLUTION AT THE CENTRE 0.117901E+01
NUMBER OF ITERATIONS AND ERROR	3	0.0000041963

STEP	4	
DETERMINANT	1.17939 E 701.0	

NUMBER OF ITERATIONS FOR PREDICTOR AND ERROR 6 ***8.0000034117***
 DERIVATIVE OF GAMA ***0.15860707*** GAMA PREDICTED ***6.754920***
 GAMA ***6.7519*** SOLUTION AT THE CENTRE ***0.121691E+01***
 NUMBER OF ITERATIONS AND ERROR 3 ***8.0000065548***

STEP 5
 DETERMINANT ***8.54178 E 700.0***
 NUMBER OF ITERATIONS FOR PREDICTOR AND ERROR 6 ***8.0000052949***
 DERIVATIVE OF GAMA ***0.12808191*** GAMA PREDICTED ***6.777479***
 GAMA ***6.7745*** SOLUTION AT THE CENTRE ***0.125521E+01***
 NUMBER OF ITERATIONS AND ERROR 3 ***8.0000031986***

STEP 6
 DETERMINANT ***5.82902 E 700.0***
 NUMBER OF ITERATIONS FOR PREDICTOR AND ERROR 6 ***8.0000085717***
 DERIVATIVE OF GAMA ***0.09815770*** GAMA PREDICTED ***6.794111***
 GAMA ***6.7912*** SOLUTION AT THE CENTRE ***0.129388E+01***
 NUMBER OF ITERATIONS AND ERROR 3 ***8.0000067286***

STEP 7
 DETERMINANT ***3.57873 E 700.0***
 NUMBER OF ITERATIONS FOR PREDICTOR AND ERROR 8 ***8.0000010482***
 DERIVATIVE OF GAMA ***0.06895831*** GAMA PREDICTED ***6.804972***
 GAMA ***6.8021*** SOLUTION AT THE CENTRE ***0.133290E+01***
 NUMBER OF ITERATIONS AND ERROR 3 ***8.0000050696***

STEP 8
 DETERMINANT ***1.72346 E 700.0***
 NUMBER OF ITERATIONS FOR PREDICTOR AND ERROR 8 ***8.0000021793***
 DERIVATIVE OF GAMA ***0.04059011*** GAMA PREDICTED ***6.810241***
 GAMA ***6.8075*** SOLUTION AT THE CENTRE ***0.137222E+01***
 NUMBER OF ITERATIONS AND ERROR 3 ***8.0000058884***

STEP 9
 DETERMINANT ***2.04348 E 699.0***
 NUMBER OF ITERATIONS FOR PREDICTOR AND ERROR 11 ***8.0000024693***
 DERIVATIVE OF GAMA ***0.01314048*** GAMA PREDICTED ***6.810109***
 GAMA ***6.8074*** SOLUTION AT THE CENTRE ***0.141182E+01***
 NUMBER OF ITERATIONS AND ERROR 3 ***8.0000040786***

STEP 10
 DETERMINANT ***-1.02974 E 700.0***
 NUMBER OF ITERATIONS FOR PREDICTOR AND ERROR 15 ***8.0000097670***
 DERIVATIVE OF GAMA ***-0.01332170*** GAMA PREDICTED ***6.804783***
 GAMA ***6.8023*** SOLUTION AT THE CENTRE ***0.145169E+01***
 NUMBER OF ITERATIONS AND ERROR 2 ***8.0000098131***

STEP 11
 DETERMINANT ***-2.02314 E 700.0***
 NUMBER OF ITERATIONS FOR PREDICTOR AND ERROR 11 ***8.0000043260***
 DERIVATIVE OF GAMA ***-0.03875039*** GAMA PREDICTED ***6.794513***
 GAMA ***6.7920*** SOLUTION AT THE CENTRE ***0.149181E+01***
 NUMBER OF ITERATIONS AND ERROR 4 ***8.0000022354***

STEP 12
 DETERMINANT ***-2.81344 E 700.0***
 NUMBER OF ITERATIONS FOR PREDICTOR AND ERROR 9 ***8.0000088576***
 DERIVATIVE OF GAMA ***-0.06309484*** GAMA PREDICTED ***6.779397***
 GAMA ***6.7770*** SOLUTION AT THE CENTRE ***0.153214E+01***
 NUMBER OF ITERATIONS AND ERROR 3 ***8.0000046262***

STEP 13
 DETERMINANT ***-3.43377 E 700.0***
 NUMBER OF ITERATIONS FOR PREDICTOR AND ERROR 9 ***8.0000053782***

DERIVATIVE OF GAMA -8.08635211 GAMA PREDICTED 6.759775
 GAMA 6.7575 SOLUTION AT THE CENTRE $0.157269E+01$
 NUMBER OF ITERATIONS AND ERROR $3 \quad 0.0000044067$

STEP 14
 DETERMINANT $-3.91183 E 700.0$
 NUMBER OF ITERATIONS FOR PREDICTOR AND ERROR $8 \quad 0.0000076642$
 DERIVATIVE OF GAMA -8.10851169 GAMA PREDICTED 6.735838
 GAMA 6.7337 SOLUTION AT THE CENTRE $0.161343E+01$
 NUMBER OF ITERATIONS AND ERROR $4 \quad 0.0000021993$

STEP 15
 DETERMINANT $-4.27135 E 700.0$
 NUMBER OF ITERATIONS FOR PREDICTOR AND ERROR $7 \quad 0.0000073910$
 DERIVATIVE OF GAMA -8.12957798 GAMA PREDICTED 6.707801
 GAMA 6.7058 SOLUTION AT THE CENTRE $0.165436E+01$
 NUMBER OF ITERATIONS AND ERROR $3 \quad 0.0000061017$

STEP 16
 DETERMINANT $-4.53249 E 700.0$
 NUMBER OF ITERATIONS FOR PREDICTOR AND ERROR $8 \quad 0.0000048660$
 DERIVATIVE OF GAMA -8.14956374 GAMA PREDICTED 6.675864
 GAMA 6.6740 SOLUTION AT THE CENTRE $0.169546E+01$
 NUMBER OF ITERATIONS AND ERROR $3 \quad 0.0000099174$

STEP 17
 DETERMINANT $-4.71230 E 700.0$
 NUMBER OF ITERATIONS FOR PREDICTOR AND ERROR $8 \quad 0.0000048233$
 DERIVATIVE OF GAMA -8.16849313 GAMA PREDICTED 6.640254
 GAMA 6.6384 SOLUTION AT THE CENTRE $0.173674E+01$
 NUMBER OF ITERATIONS AND ERROR $3 \quad 0.0000072041$

STEP 18
 DETERMINANT $-4.82517 E 700.0$
 NUMBER OF ITERATIONS FOR PREDICTOR AND ERROR $8 \quad 0.0000046367$
 DERIVATIVE OF GAMA -8.18639272 GAMA PREDICTED 6.601165
 GAMA 6.5995 SOLUTION AT THE CENTRE $0.177818E+01$
 NUMBER OF ITERATIONS AND ERROR $4 \quad 0.0000022164$

STEP 19
 DETERMINANT $-4.88325 E 700.0$
 NUMBER OF ITERATIONS FOR PREDICTOR AND ERROR $8 \quad 0.0000045085$
 DERIVATIVE OF GAMA -8.20329586 GAMA PREDICTED 6.558802
 GAMA 6.5572 SOLUTION AT THE CENTRE $0.181979E+01$
 NUMBER OF ITERATIONS AND ERROR $4 \quad 0.0000031741$

STEP 20
 DETERMINANT $-4.89673 E 700.0$
 NUMBER OF ITERATIONS FOR PREDICTOR AND ERROR $8 \quad 0.0000044371$
 DERIVATIVE OF GAMA -8.21923678 GAMA PREDICTED 6.513343
 GAMA 6.5118 SOLUTION AT THE CENTRE $0.186156E+01$
 NUMBER OF ITERATIONS AND ERROR $4 \quad 0.0000033500$

B. PROGRAM DIFEQ AND TEST RESULTS

```

PROGRAM DIFEQ
C
--diffusion equation in two dimensions with variable coefficients
  inversion of the spectral operator accelerated by incomplete
  inverse of the finite difference matrix ( centered
  differences on Chebyshev mesh )
- variable order time integration ( order 2 to 6 )
  lowest order is trapezoidal scheme
  orders 3 to 6 are Gear's scheme
ratio of time increments must be larger than ORDER-2
incomplete Crout decomposition with 15 diagonals
Q is source, G diffusion coefficient, H solution
Q(x,y)=-2D(1+x+C(1+CF1y))exp(x+CF1y)+2(C+1)CF2
G(x,y)=Dexp(x+CF1y)-CF2
H(t,x,y)=exp(-ALFAt+BETAx+GAMAY)+xx+Cyy

PARAMETER(IX=16,IZ=16,IXORDER=6)
PARAMETER(NX2=IX+2,NZ2=IZ+1,IXORD=2*IXORDER-1)
PARAMETER(IX1=IX-1,IZ1=IZ-1)
PARAMETER(NNX=IX1*IZ1)
COMMON /FT/NPTS,NSKIP,MTRAN,MSKIP,ISIGN,LOG,IEX
COMMON /EXS/NMAX,EX(4096)
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
COMMON /PARAM/ALFA,BETA,GAMA,CC,DD,CF1,CF2
COMMON /PARAM1/PI,DX,DZ,RT2,DT,DXT,DXX
COMMON /SIZE/BOXX,BOXZ,BOXH,BOZH,PIX,PIZ
COMMON /GRID/XX(65),ZZ(65)
REAL X1(NX2,NZ2),X2(NX2,NZ2),U1(NX2,NZ2),BMTX(NNX,13)
REAL U2(NX2,NZ2),P(NX2,NZ2),AP(NX2,NZ2),R(IX1,IZ1)
REAL X(NX2,NZ2),DIFF(NX2,NZ2),A(NX2,NX2)
REAL Y(NX2,NZ2),U(NX2,NZ2,IXORD),AMTX(NNX,5),W(NX2,NZ2)
REAL ALF(6,6),BET(6),R1(IX1,IZ1),R2(IX1,IZ1)
INTEGER KST(6)
TIME=0.1
DTMAX=0.01
ERR=1.E-8
LMAX=200
IXORD1=IXORDER-1
IXORD2=IXORDER-2
RATIO=10.
RTOR=RATIO**IXORD2
KRATIO=IFIX(RATIO)
DO 10 I=1,IXORD1
  KST(I)=I*KRATIO
10 C
  - set parameters of diffusion coefficient, source and solution
  ALFA=-1.
  BETA=-3.
  GAMA=BETA
  DD=1.
  CC=0.
  CF1=-(BETA*BETA+GAMA*GAMA+BETA)/GAMA
  CF2=ALFA/(BETA*BETA+GAMA*GAMA)
  WRITE(6,20)ALFA,BETA,GAMA,CC,DD,CF1,CF2
20  FORMAT(/,* ALFA*,E9.2,* BETA*,E9.2,* GAMA*,E9.2,* CC*,E9.2,
  1* DD*,E9.2,* CF1*,E9.2,* CF2*,E9.2,/)
```

```

30    WRITE(6,30)TIME,DTMAX,RATIO,S,LMAX,ERR
      FORMAT(/,* TIME *,E16.6,* DTMAX *,E16.4,* RATIO *,E16.4,
2* S *,E12.6,* LMAX *,I8,* ERR *,E18.18,/)

      NN=NNX
      NX=IX
      NZ=IZ
      CALL SETGS(BET,ALF)
      CALL EXSET(EX,NMAX)
      CALL SETMAT(NN,AMTX)
      CALL DCS(U1,DIFF,A)
      C   - set initial condition
      DT=DTMAX/RTOR
      S=0.95
      DO 50 J=1,NZP
      ZZ1=ZZ(J)
      DO 40 I=1,NXP
      XX1=XX(I)
      W(I,J)=H(0.,XX1,ZZ1)
      U(I,J,1)=W(I,J)
      U(I,J,IXORD)=W(I,J)
      40  CONTINUE
      50  TT=0.
      C   -- solution at the time TT is stored in W(I,J)
      C   solutions at previous times are stored in U(I,J,K)
      KL=0
      L=0
      IORDER=1
      C   ---- order loop -----
      1111 IORDER=IORDER+1
      IORD1=IORDER-1
      IORDP1=IORDER+1
      DTB=DT*BET(IORDER)
      DXT=DXX/DTB
      DXT1=1./DXT
      DXTX=DXT1*DXT1
      DXT12=2.*DXT1
      DXD=DXX/BET(IORDER)
      DXT2=2.*DXT
      CALL CR(S,NN,NXM,AMTX,BMTX)
      C   ---- time loop -----
      999  L=L+1
      TT=TT+DT
      IF(TT.GE.TIME)GOTO 666
      C   -- if IORDER is 2 start with trapezoidal scheme
      IF(IORDER.GT.2)GOTO 444
      CALL PRA(X1,X2,U1,U2,DIFF,W,AP)
      DO 90 J=2,NZ
      DO 80 I=2,NX
      80  W(I,J)=3.*W(I,J)-DXT12*AP(I,J)+DT*A(I,J)
      90  CONTINUE
      DO 130 J=1,NZP
      DO 120 I=1,NXP
      120 Y(I,J)=DXT2*U(I,J,1)-AP(I,J)+DXD*A(I,J)
      130 CONTINUE
      GOTO 555
      C   -- otherwise with Gear method
      444  DO 170 J=1,NZP
      DO 160 I=1,NXP
      160 Y(I,J)=0.
      170 CONTINUE
      DO 250 KO=1,IORDER
      CX=DXT*ALF(KO,IORDER)
      DO 240 J=1,NZP
      DO 230 I=1,NXP

```

```

230 Y(I,J)=Y(I,J)+CX*U(I,J,K0)
240 CONTINUE
250 CONTINUE
CALL PRA(X1,X2,U1,U2,DIFF,Y,AP)
DO 270 J=2,NZ
DO 260 I=2,NX
260 W(I,J)=DXTI2*Y(I,J)-DXTX*AP(I,J)+DTB*A(I,J)
270 CONTINUE
DO 290 J=1,NZP
DO 280 I=1,NXP
280 Y(I,J)=Y(I,J)+DXX*A(I,J)
290 CONTINUE
555 DO 340 I=1,NXP
XX1=XX(I)
W(I,1)=H(TT,XX1,BOZH)
340 W(I,NZP)=H(TT,XX1,-BOZH)
DO 350 I=1,NZP
ZZ1=ZZ(I)
W(I,I)=H(TT,BOXH,ZZ1)
350 W(NXP,I)=H(TT,-BOXH,ZZ1)
CALL CG(NN,AMTX,BMTX,X1,X2,U1,U2,R,R1,R2,P,AP,DIFF
1,LMAX,ERR,L1,ERR1,Y,W)
KL=KL+L1
IF(ERR1.GT.ERR)GOTO 666
DO 460 IK=1,IORD1
IK1=IORDP1-IK
IK2=IK1-1
DO 450 J=1,NZP
DO 440 I=1,NXP
440 U(I,J,IK1)=U(I,J,IK2)
450 CONTINUE
460 CONTINUE
DO 540 J=1,NZP
DO 530 I=1,NXP
530 U(I,J,1)=W(I,J)
540 CONTINUE
IF(IORDER.EQ.IXORDER)GOTO 999
DO 550 KKJ=1,IORDER
IF(L.EQ.KST(KKJ))GOTO 888
550 CONTINUE
GOTO 999
888 KX=IXORD-KKJ
DO 590 J=1,NZP
DO 580 I=1,NXP
580 U(I,J,KX)=W(I,J)
590 CONTINUE
IF(L.EQ.KST(IORDER))GOTO 222
GOTO 999
222 L=L/KRATIO
IORX=IXORD-IORDP1
DO 750 IJK=2,IORDP1
IKK=IORX+IJK
DO 730 J=1,NZP
DO 720 I=1,NXP
720 U(I,J,IJK)=U(I,J,IKK)
730 CONTINUE
750 CONTINUE
DT=RATIO*DT
GOTO 1111
666 TT=TT-DT
WRITE(6,420)KL,L1,ERR1,IORDER
420 FORMAT(1,*ITER*,2(I8),*ERR1*,E14.6,*ORDER *,I6,/)

CALL SOL(TT,AP,W)
STOP

```

```

END
SUBROUTINE PRA(X1,X2,U1,U2,DIFF,Y,X)
COMMON /FT/NPTS,NSKIP,MTRAN,MSKIP,ISIGN,LOG,IEX
COMMON /EXS/NMAX,EX(4096)
COMMON /SIZE/BOXX,BOXZ,BOXH,BOZH,PIX,PIZ
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
COMMON /PARAM/ALFA,BETA,GAMA,CC,D,CF1,CF2
COMMON /PARAM1/PI,DX,DZ,RT2,DT,DXT,DXX
REAL X1(NXPP,1),X2(NXPP,1),U1(NXPP,1),U2(NXPP,1)
REAL X(NXPP,1),Y(NXPP,1),DIFF(NXPP,1),WORK(8192)

100  DO 110 J=1,NZP
110  DO 100 I=1,NXP
      X2(I,J)=Y(I,J)
      XI(I,J)=Y(I,J)
CONTINUE
CALL XDERIV(X1,U1,WORK)
CALL ZDERIV(X2,U2,WORK)
DO 200 J=1,NZP
DO 190 I=1,NXP
      U2(I,J)=U2(I,J)*DIFF(I,J)
      U1(I,J)=U1(I,J)*DIFF(I,J)
190 CONTINUE
CALL XDERIV(U1,X1,WORK)
CALL ZDERIV(U2,X2,WORK)
DO 270 J=1,NZP
DO 260 I=1,NXP
      X(I,J)=DXT*Y(I,J)-DXX*(X2(I,J)+X1(I,J))
260 CONTINUE
270 RETURN
END

SUBROUTINE SETGS(BET,ALF)
REAL BET(6),ALF(6,6)
C   --sets Gear coefficients ( BET(2) is coefficient for trapezoidal
      scheme ) and FFT parameters
COMMON /PARAM1/PI,DX,DZ,RT2,DT,DXT,DXX
COMMON /SIZE/BOXX,BOXZ,BOXH,BOZH,PIX,PIZ
COMMON /EXS/NMAX,EX(4096)
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
COMMON /FT/NPTS,NSKIP,MTRAN,MSKIP,ISIGN,LOG,IEX
COMMON /GRID/XX(65),ZZ(65)
PI=4.*ATAN(1.)
BET(2)=0.5
BET(3)=6./11.
BET(4)=12./25.
BET(5)=60./137.
BET(6)=60./147.
C   - first index is for summation and second for order
ALF(1,3)=18./11.
ALF(1,4)=48./25.
ALF(1,5)=300./137.
ALF(1,6)=360./147.
ALF(2,3)=-9./11.
ALF(2,4)=-36./25.
ALF(2,5)=-300./137.
ALF(2,6)=-450./147.
ALF(3,3)=2./11.
ALF(3,4)=16./25.
ALF(3,5)=200./137.
ALF(3,6)=400./147.
ALF(4,4)=-3./25.
ALF(4,5)=-75./137.
ALF(4,6)=-225./147.
ALF(5,5)=12./137.
ALF(5,6)=72./147.

```

```

C ALF(6,6)=-10./147.
C -set FFT parameters
MAXNP=NX+1
NXM=NX-1
NXP=NX+1
NZP=NZ+1
NZM=NZ-1
NXPP=NX+2
NN=NXP*NZP
NTOT=NXPP*NZP
NPTS=2*NX
NMAX=NPTS
NSKIP=1
MTRAN=NZP
MSKIP=NXPP/2
LOG=LOG2(NPTS)
IEX=NMAX/NPTS
BOXX=2.
BOXZ=2.
DX=1./FLOAT(NX)
DZ=1./FLOAT(NZ)
DXX=DX*DX
PIX=PI*DX
PIZ=PI*DZ
BOZH=BOXZ/2.
BOXH=BOXX/2.
20 WRITE(6,20)NX,NZ,DX,DZ
FORMAT(1,* NX*,I4,* NZ*,I5,* DX DZ*,2(E9.2),/)
DO 40 J=1,NZP
40 ZZ(J)=COS(PIZ*FLOAT(J-1))*BOZH
DO 60 I=1,NXP
60 XX(I)=COS(PIX*FLOAT(I-1))*BOXH
RETURN
END
SUBROUTINE XDERIV(A,B,WORK)
COMMON /FT/NPTS,NSKIP,MTRAN,MSKIP,ISIGN,LOG,IEX
COMMON /EXS/NMAX,EX(1)
COMMON /CHEBRL/NCHEB,NSKCHEB,MCHEB,MSKCHEB,FZ
COMMON /SIZE/BOXX,BOXZ,BOXH,BOZH,PIX,PIZ
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
REAL A(NXPP,1),B(NXPP,1),WORK(8192)
DO 20 J=1,NZP
DO 10 I=1,NXPP
10 B(I,J)=A(I,J)
CONTINUE
NPTS=2*NX
NSKIP=1
MTRAN=NZP
MSKIP=NXPP/2
ISIGN=-1
LOG=LOG2(NPTS)
IEX=NMAX/NPTS
NCHEB=NX
NSKCHEB=1
MCHEB=NZP
MSKCHEB=NXPP
FZ=1.0/(BOXX*2.*FLOAT(NX))
CALL SRR(B,EX,WORK)
CALL RLCHBDF(B,B)
ISIGN=1
CALL SRR(B,EX,WORK)
RETURN
END
SUBROUTINE ZDERIV(A,B,WORK)

```

```

COMMON /FT/NPTS,NSKIP,MTRAN,MSKIP,ISIGN,LOG,IEX
COMMON /EXS/NMAX,EX(1)
COMMON /CHEB/NCHEB,NSKCHEB,MCHEB,MSKCHEB,FZ
COMMON /SIZE/BOXX,BOXZ,BOXH,BOZH,PIX,PIZ
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
REAL A(NXPP,1),B(NXPP,1),WORK(8192)
DO 20 J=1,NZP
DO 10 I=1,NXPP
B(I,J)=A(I,J)
CONTINUE
NPTS=2*NZ
NXHP=NXPP/2
NSKIP=NXHP
MTRAN=NSKIP
MSKIP=1
ISIGN=-1
LOG=LOG2(NPTS)
IEX=NMAX/NPTS
NCHEB=NZ
NSKCHEB=NXHP
MCHEB=NXHP
MSKCHEB=1
FZ=1.0/(BOXZ*2.*FLOAT(NZ))
CALL SCSC(B,EX,WORK)
CALL CHEBDIF(B,B)
ISIGN=1
CALL SCSC(B,EX,WORK)
RETURN
END
FUNCTION Q(X,Y)
COMMON /PARAM/ALFA,BETA,GAMA,C,D,CF1,CF2
Q=-2.*D*(1.+X+C*(1.+CF1*Y))*EXP(X+CF1*Y)+2.*(C+1.)*CF2
RETURN
END
FUNCTION H(T,X,Y)
COMMON /PARAM/ALFA,BETA,GAMA,C,D,CF1,CF2
H=EXP(-ALFA*T+BETA*X+GAMA*Y)+X*X+C*Y*Y
RETURN
END
FUNCTION G(X,Y)
COMMON /PARAM/ALFA,BETA,GAMA,C,D,CF1,CF2
G=D*EXP(X+CF1*Y)-CF2
RETURN
END
FUNCTION FAP(Z)
COMMON /PARAM/PI,DX,DZ,RT2,DT,DXT,DXX
FAP=1.0/(PI*SQRT(1.-Z*Z))
RETURN
END
SUBROUTINE CG(NN,AMTX,BMTX,X1,X2,U1,U2,R,R1,R2,P,AP,DIFF
1,LMAX,ERR,L,ERR1,Y,X)
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
REAL X(NXPP,1),Y(NXPP,1),U1(NXPP,1)
REAL AP(NXPP,1),R(NXM,1),P(NXPP,1),AMTX(NN,5)
REAL X1(NXPP,1),X2(NXPP,1),DIFF(NXPP,1),U2(NXPP,1)
REAL BMTX(NN,13),R1(NXM,1),R2(NXM,1)
CALL PRA(X1,X2,U1,U2,DIFF,X,AP)
DO 20 J=2,NZ
DO 10 I=2,NX
R(I-1,J-1)=Y(I,J)-AP(I,J)
CONTINUE
CALL INV(NN,NXM,AMTX,BMTX,R,R1,R2)
RR=0.
DO 40 J=1,NZM

```

```

48     RR=RR+SDOT(NXM,R(1,J),1,R2(1,J),1)
      CONTINUE
      DO 80 J=1,NZM
      DO 70 I=1,NXM
      P(I+1,J+1)=R2(I,J)
      CONTINUE
      L=0
      999 L=L+1
      CALL PRA(X1,X2,U1,U2,DIFF,P,AP)
      PMP=0.
      DO 110 J=2,NZ
      PMP=PMP+SDOT(NXM,P(2,J),1,AP(2,J),1)
      CONTINUE
      PMP=RR/PMP
      R1R=RR
      DO 140 J=2,NZ
      CALL SAXPY(NXM,PMP,P(2,J),1,X(2,J),1)
      CALL SAXPY(NXM,-PMP,AP(2,J),1,R(1,J-1),1)
      CONTINUE
      CALL INV(NN,NXM,AMTX,BMTX,R,R1,R2)
      RR=0.
      DO 160 J=1,NZM
      RR=RR+SDOT(NXM,R(1,J),1,R2(1,J),1)
      CONTINUE
      ERR1=SQRT(ABS(RR))
      IF(L.GE.LMAX)RETURN
      IF(ERR1.LE.ERR)RETURN
      PMP=RR/R1R
      DO 220 J=2,NZ
      DO 210 I=2,NX
      P(I,J)=R2(I-1,J-1)+PMP*P(I,J)
      CONTINUE
      220 GOTO 999
      END
      SUBROUTINE SETMAT(NN,AMTX)
      COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
      COMMON /SIZE/BOXX,BOZZ,BOXH,BOZH,PIX,PIZ
      COMMON /GRID/XX(65),ZZ(65)
      COMMON /PARAM1/PI,DX,DZ,RT2,DT,DXT,DXX
      REAL AMTX(NN,5)
      DO 50 J=1,NZM
      J1=(J-1)*NXM
      YF=COS(PIZ*(FLOAT(J)+0.5))*BOZH
      YB=COS(PIZ*(FLOAT(J)-0.5))*BOZH
      YFB=ZZ(J+1)
      Y1=FAP(YB)
      Y2=0.5*FAP(YFB)
      Y3=FAP(YF)
      Y21=Y2*Y1
      Y23=Y3*Y2
      DO 40 I=1,NXM
      XF=COS(PIX*(FLOAT(I)+0.5))*BOXH
      XB=COS(PIX*(FLOAT(I)-0.5))*BOXH
      XFB=XX(I+1)
      X1=FAP(XB)
      X2=0.5*FAP(XFB)
      X3=FAP(XF)
      X21=X2*X1
      X23=X2*X3
      AMTX(I+J1,5)=-(G(XF,YF)+G(XB,YF))*Y23
      AMTX(I+J1,1)=-(G(XF,YB)+G(XB,YB))*Y21
      AMTX(I+J1,4)=-(G(XF,YF)+G(XF,YB))*X23
      AMTX(I+J1,2)=-(G(XB,YF)+G(XB,YB))*X21
      40 CONTINUE
      50

```

```

DO 90 J=1,NZM
J1=(J-1)*NXM
DO 80 I=1,NXM
80 AMTX(I+J1,3)=-AMTX(I+J1,5)-AMTX(I+J1,1)
I-AMTX(I+J1,4)-AMTX(I+J1,2)
90 CONTINUE
DO 110 I=1,NN
AMTX(I,2)=AMTX(I+1,2)
110 AMTX(I,1)=AMTX(I+NXM,1)
DO 130 I=NXM,NN,NXM
AMTX(I,4)=0.
130 AMTX(I,2)=0.
JN=NN-NXM
DO 150 I=1,NXM
AMTX(I+JN,1)=0.
150 AMTX(I+JN,5)=0.
RETURN
END
SUBROUTINE DCS(DG,DIFF,A)
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
COMMON /SIZE/B0XX,B0XZ,B0XH,B0ZH,PIX,PIZ
COMMON /GRID/XXX(65),ZZ(65)
REAL DIFF(NXPP,NZP),A(NXPP,NZP),DG(1)
NN=NXM*NZM
DO 50 J=1,NZP
ZZ1=ZZ(J)
DO 40 I=1,NXP
XX1=XX(I)
A(I,J)=Q(XX1,ZZ1)
40 DIFF(I,J)=G(XX1,ZZ1)
50 CONTINUE
DO 70 J=1,NZP
J1=(J-1)*NXP
DO 60 I=1,NXP
DG(I+J1)=DIFF(I,J)
60 CONTINUE
KMX=ISMAX(NN,DG,1)
KMN=ISMIN(NN,DG,1)
DIFMAX=DG(KMX)
DIFMIN=DG(KMN)
IF(DIFMIN.LE.0.0)GOTO 111
RRD=DIFMAX/DIFMIN
111 170 WRITE(6,170)DIFMAX,DIFMIN,RRD
FORMAT(/,* DIFMAX DIFMIN *,2(E14.6),* RATIO *,E14.6,/)

170 RETURN
END
SUBROUTINE SOL(TT,AP,W)
COMMON /PARAM/PI,DX,DZ,RT2,DT,DXT,DXX
COMMON /SIZE/BOXX,B0XZ,B0XH,B0ZH,PIX,PIZ
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
COMMON /GRID/XXX(65),ZZ(65)
REAL AP(NXPP,NZP),W(NXPP,NZP)
DO 30 J=1,NZP
ZZ1=ZZ(J)
DO 10 I=1,NXP
XX1=XX(I)
AP(I,J)=H(TT,XX1,ZZ1)
10 CONTINUE
ER=0.
30 DO 780 J=1,NZP
DO 770 I=1,NXP
770 ER=ER+ABS(AP(I,J)-W(I,J))
780 CONTINUE
ER=ER*DX*DZ

```

```

      WRITE(6,880)TT,ER
880  FORMAT(//, * TIME *,E16.6,* ERROR*,E16.6,/)
      RETURN
      END
      SUBROUTINE CR(S,N,M,AMTX,BMTX)
      COMMON /PARAM1/PI,DX,DZ,RT2,DT,DXT,DXX
      REAL AMTX(N,5),BMTX(N,13)
      BMTX(1,7)=S*(AMTX(1,3)+DXT)
      BMTX(1,6)=AMTX(1,2)
      BMTX(1,8)=AMTX(1,4)
      AMTX(1,5)=AMTX(1,6)
      BMTX(2,7)=S*(AMTX(2,3)+DXT)-BMTX(1,6)*BMTX(1,8)/BMTX(1,7)
      BMTX(2,6)=AMTX(2,2)-BMTX(1,5)*BMTX(1,8)/BMTX(1,7)
      BMTX(2,8)=AMTX(2,4)-BMTX(1,6)*BMTX(1,9)/BMTX(1,7)
      BMTX(2,4)=-BMTX(1,3)*BMTX(1,8)/BMTX(1,7)
      BMTX(2,10)=-BMTX(1,6)*BMTX(1,11)/BMTX(1,7)
      BMTX(2,3)=-BMTX(1,2)*BMTX(1,8)/BMTX(1,7)
      BMTX(2,11)=-BMTX(1,6)*BMTX(1,12)/BMTX(1,7)
      BMTX(2,2)=-BMTX(1,1)*BMTX(1,8)/BMTX(1,7)
      BMTX(2,12)=-BMTX(1,6)*BMTX(1,13)/BMTX(1,7)
      BMTX(2,1)=-AMTX(1,1)*BMTX(1,8)/BMTX(1,7)
      BMTX(2,13)=-BMTX(1,6)*AMTX(1,5)/BMTX(1,7)
      DO 20 I=3,M
      BMTX(I,7)=S*(AMTX(I,3)+DXT)-BMTX(I-1,6)*BMTX(I-1,8)
1/BMTX(I-1,7)-BMTX(I-2,5)*BMTX(I-2,9)/BMTX(I-2,7)
      BMTX(I,6)=AMTX(I,2)-BMTX(I-1,5)*BMTX(I-1,8)/BMTX(I-1,7)
      BMTX(I,8)=AMTX(I,4)-BMTX(I-1,6)*BMTX(I-1,9)/BMTX(I-1,7)
      BMTX(I,4)=-BMTX(I-1,3)*BMTX(I-1,8)/BMTX(I-1,7)
1-BMTX(I-2,2)*BMTX(I-2,9)/BMTX(I-2,7)
      BMTX(I,10)=-BMTX(I-1,6)*BMTX(I-1,11)/BMTX(I-1,7)
1-BMTX(I-2,5)*BMTX(I-2,12)/BMTX(I-2,7)
      BMTX(I,3)=-BMTX(I-1,2)*BMTX(I-1,8)/BMTX(I-1,7)
1-BMTX(I-2,1)*BMTX(I-2,9)/BMTX(I-2,7)
      BMTX(I,11)=-BMTX(I-1,6)*BMTX(I-1,12)/BMTX(I-1,7)
1-BMTX(I-2,5)*BMTX(I-2,13)/BMTX(I-2,7)
      BMTX(I,2)=-BMTX(I-1,1)*BMTX(I-1,8)/BMTX(I-1,7)
1-AMTX(I-2,1)*BMTX(I-2,9)/BMTX(I-2,7)
      BMTX(I,12)=-BMTX(I-1,6)*BMTX(I-1,13)/BMTX(I-1,7)
1-BMTX(I-2,5)*AMTX(I-2,5)/BMTX(I-2,7)
      BMTX(I,1)=AMTX(I-1,1)*BMTX(I-1,8)/BMTX(I-1,7)
      BMTX(I,13)=-BMTX(I-1,6)*AMTX(I-1,5)/BMTX(I-1,7)
      CONTINUE
20   DO 50 I=M+1,N
      BMTX(I,7)=S*(AMTX(I,3)+DXT)-BMTX(I-1,6)*BMTX(I-1,8)
1/BMTX(I-1,7)-BMTX(I-2,5)*BMTX(I-2,9)/BMTX(I-2,7)
1-BMTX(I-M+4,4)*BMTX(I-M+4,10)/BMTX(I-M+4,7)
2-BMTX(I-M+3,3)*BMTX(I-M+3,11)/BMTX(I-M+3,7)
2-BMTX(I-M+2,2)*BMTX(I-M+2,12)/BMTX(I-M+2,7)
3-BMTX(I-M+1,1)*BMTX(I-M+1,13)/BMTX(I-M+1,7)
3-AMTX(I-M,1)*AMTX(I-M,5)/BMTX(I-M,7)
      BMTX(I,6)=AMTX(I,2)-BMTX(I-1,5)*BMTX(I-1,8)/BMTX(I-1,7)
1-BMTX(I-M+4,3)*BMTX(I-M+4,10)/BMTX(I-M+4,7)
1-BMTX(I-M+3,2)*BMTX(I-M+3,11)/BMTX(I-M+3,7)
2-BMTX(I-M+2,1)*BMTX(I-M+2,12)/BMTX(I-M+2,7)
2-AMTX(I-M+1,1)*BMTX(I-M+1,13)/BMTX(I-M+1,7)
      BMTX(I,8)=AMTX(I,4)-BMTX(I-1,6)*BMTX(I-1,9)/BMTX(I-1,7)
1-BMTX(I-M+4,4)*BMTX(I-M+4,11)/BMTX(I-M+4,7)
1-BMTX(I-M+3,3)*BMTX(I-M+3,12)/BMTX(I-M+3,7)
2-BMTX(I-M+2,2)*BMTX(I-M+2,13)/BMTX(I-M+2,7)
2-BMTX(I-M+1,1)*AMTX(I-M+1,5)/BMTX(I-M+1,7)
      BMTX(I,5)=-BMTX(I-M+4,2)*BMTX(I-M+4,10)/BMTX(I-M+4,7)
1-BMTX(I-M+3,1)*BMTX(I-M+3,11)/BMTX(I-M+3,7)
1-AMTX(I-M+2,1)*BMTX(I-M+2,12)/BMTX(I-M+2,7)
      BMTX(I,9)=-BMTX(I-M+4,4)*BMTX(I-M+4,12)/BMTX(I-M+4,7)

```

```

2-BMTX(I-M+3,3)*BMTX(I-M+3,13)/BMTX(I-M+3,7)
2-BMTX(I-M+2,2)*AMTX(I-M+2,5)/BMTX(I-M+2,7)
BMTX(I,4)=-BMTX(I-1,3)*BMTX(I-1,8)/BMTX(I-1,7)
1-BMTX(I-2,2)*BMTX(I-2,9)/BMTX(I-2,7)
BMTX(I,10)=-BMTX(I-1,6)*BMTX(I-1,11)/BMTX(I-1,7)
1-BMTX(I-2,5)*BMTX(I-2,12)/BMTX(I-2,7)
BMTX(I,3)=-BMTX(I-1,2)*BMTX(I-1,8)/BMTX(I-1,7)
1-BMTX(I-2,1)*BMTX(I-2,9)/BMTX(I-2,7)
BMTX(I,11)=-BMTX(I-1,6)*BMTX(I-1,12)/BMTX(I-1,7)
1-BMTX(I-2,5)*BMTX(I-2,13)/BMTX(I-2,7)
BMTX(I,2)=-BMTX(I-1,1)*BMTX(I-1,8)/BMTX(I-1,7)
1-AMTX(I-2,1)*BMTX(I-2,9)/BMTX(I-2,7)
BMTX(I,12)=-BMTX(I-1,6)*BMTX(I-1,13)/BMTX(I-1,7)
1-BMTX(I-2,5)*AMTX(I-2,5)/BMTX(I-2,7)
BMTX(I,1)=-AMTX(I-1,1)*BMTX(I-1,8)/BMTX(I-1,7)
BMTX(I,13)=-BMTX(I-1,6)*AMTX(I-1,5)/BMTX(I-1,7)
50 CONTINUE
RETURN
END
SUBROUTINE INV(N,M,AMTX,BMTX,Y,R,X)
REAL AMTX(N,5),BMTX(N,13),R(N),X(N),Y(N)
R(1)=Y(1)/BMTX(1,7)
R(2)=(Y(2)-BMTX(1,6)*R(1))/BMTX(2,7)
DO 40 I=3,M
R(I)=(Y(I)-BMTX(I-1,6)*R(I-1)-BMTX(I-2,5)*R(I-2))
3/BMTX(I,7)
40 CONTINUE
DO 50 I=M+1,N
R(I)=(Y(I)-BMTX(I-1,6)*R(I-1)-BMTX(I-2,5)*R(I-2)
1-BMTX(I-M+4,4)*R(I-M+4)-BMTX(I-M+3,3)*R(I-M+3)
2-BMTX(I-M+2,2)*R(I-M+2)-BMTX(I-M+1,1)*R(I-M+1)
3-AMTX(I-M,1)*R(I-M))/BMTX(I,7)
50 CONTINUE
DO 60 I=1,M
X(N+1-I)=R(N+1-I)-(BMTX(N+1-I,8)*X(N+2-I)+BMTX(N+1-I,9)*X(N+3-I)
1+BMTX(N+1-I,10)*X(N+1+M-4-I)+BMTX(N+1-I,11)*X(N+1+M-3-I)
2+BMTX(N+1-I,12)*X(N+1+M-2-I)+BMTX(N+1-I,13)*X(N+1+M-1-I)
3/BMTX(N+1-I,7)
60 CONTINUE
DO 70 I=M+1,N
X(N+1-I)=R(N+1-I)-(BMTX(N+1-I,8)*X(N+2-I)+BMTX(N+1-I,9)*X(N+3-I)
1+BMTX(N+1-I,10)*X(N+1+M-4-I)+BMTX(N+1-I,11)*X(N+1+M-3-I)
2+BMTX(N+1-I,12)*X(N+1+M-2-I)+BMTX(N+1-I,13)*X(N+1+M-1-I)
3+AMTX(N+1-I,5)*X(N+1+M-I))/BMTX(N+1-I,7)
70 CONTINUE
RETURN
END

```

Diffusion coefficient with gaussian profile

```

a(x,y)=10000exp[-6(xx+yy)]
ALFA=0.60E+01 BETA 0.10E+01 GAMA 0.10E+01 CC 0.10E+01 DD 0.10E+05
TIME 0.10E+00 RATIO 0.10E+02 S 0.130E+01 LMAX 200 ERR 0.10E-04
NX 16 NZ 16 DX DZ 0.62E-01 0.62E-01
DIFMAX DIFMIN 0.10E+05 0.614421E-01 RATIO 0.162755E+06
DT 0.20E-02
ITER 4958 39 ERR1 0.928657E-05 ORDER 6
TIME 0.10E+00 SOLUTION IN THE CENTRE 0.1606270952E+01

```

```

DT 0.40E-02
ITER 4607 66 ERR1 0.853160E-05 ORDER 6
TIME 0.10E+00 SOLUTION IN THE CENTRE 0.1606270952E+01

```

C. PROGRAM CRYSTALG AND TEST RESULTS

PROGRAM CRYSTALG

```
-- Diffusion equation
domain of integration is divided in IDOM (=4) equal
subdomains along x-axis

boundary conditions are: Neumann on the top and bottom
of second and third subdomain
Dirichlet everywhere else

Q is source, G diffusion coefficient, H solution
Q(x,y)=-C(12yy-4)(exp(-BETAX)+CF1)
G(x,y)=exp(-BETAX)+CF1
H(t,x,y)=exp(ALFAt+BETAX)+C(yy-1)(yy-1)

CF1=ALFA/(BETA*BETA)

PARAMETER(IX=8,IZ=8,IXORDER=4,IDOM=4)
PARAMETER(NX2=IX+2,NZ2=IZ+1,IXORD=2*IXORDER-1)
COMMON /FT/NPTS,NSKIP,MTRAN,MSKIP,ISIGN,LOG,IEX
COMMON /EXS/NMAX,EX(4096)
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
COMMON /PARAM/ALFA,BETA,CC,CF1
COMMON /PARAM1/PI,DX,DZ,RT2,DT,DXT,DXX
COMMON /SIZE/BOXX,BOXZ,BOXH,BOZH,PIX,PIZ
COMMON /GRID/XX(65),ZZ(65)
COMMON /DELS/DDL(65),DDR(65),DELL,DELR
REAL X1(NX2,NZ2),X2(NX2,NZ2),P(NX2,NZ2,IDOM),AP(NX2,NZ2,IDOM)
2,X(NX2,NZ2,IDOM),DIFF(NX2,NZ2,IDOM),A(NX2,NZ2,IDOM)
3,Y(NX2,NZ2,IDOM),U(NX2,NZ2,IXORD,IDOM),D(NX2,NZ2,IDOM)
4,AMTX(NX2,NZ2,5,IDOM),W(NX2,NZ2,IDOM),R(NX2,NZ2,IDOM)
REAL ALF(6,6),BET(6)
INTEGER KST(6)
TIME=0.1
DTMAX=0.01
S=1.1
ERR=1.E-5
LMAX=100
KD=IDOM
IXORD1=IXORDER-1
IXORD2=IXORDER-2
RATIO=10.
RTOR=RATIO**IXORD2
DT=DTMAX/RTOR
KRATIO=IFIX(RATIO)
DO 10 I=1,IXORD1
KST(I)=I*KRATIO
10 - set parameters of diffusion coefficient, source and solution
ALFA=1.
BETA=-1.
CC=1.
CF1=ALFA/(BETA*BETA)
WRITE(6,20)ALFA,BETA,CC,CF1
20 FORMAT(/,* ALFA*,E9.2,* BETA*,E9.2,* CC*,E9.2,* CF1*,E9.2,/)
WRITE(6,30)TIME,DTMAX,RATIO,S,LMAX,ERR
30 FORMAT(/,* TIME *,E16.6,* DTMAX *,E16.4,* RATIO *,E16.4,
2* S *,E12.6,* LMAX *,I8,* ERR *,E18.10,/)
NX=IX
```

```

NZ=IZ
CALL SETGS(BET,ALF)
CALL EXSET(EX,NMAX)
CALL SETMAT(KD,AMTX)
CALL DCS(KD,X1,DIFF,A)
C      - set initial condition
DO 60 K=1,KD
DO 50 J=1,NZP
ZZ1=ZZ(J)
DO 40 I=1,NXP
XX1=XX(I)
W(I,J,K)=H(K,0.,XX1,ZZ1)
U(I,J,1,K)=W(I,J,K)
40 U(I,J,IXORD,K)=W(I,J,K)
50 CONTINUE
60 CONTINUE
TT=0.
C      -- solution at the time TT is stored in W(I,J)
C      solutions at previous times are stored in U(I,J,K)
KL=0
L=0
IORDER=1
C      ---- order loop -----
1111 IORDER=IORDER+1
IORD1=IORDER-1
IORDP1=IORDER+1
DTB=DT*BET(IORDER)
DXT=DXX/DTB
DXT1=1./DXT
DXTX=DXT1*DXT1
DXT12=2.*DXT1
DXD=DXX/BET(IORDER)
DXT2=2.*DXT
CALL CR(KD,S,AMTX,D)
C      ---- time loop -----
999 L=L+1
TT=TT+DT
IF(TT.GE.TIME)GOTO 666
C      -- if IORDER is 2 start with trapezoidal scheme
IF(IORDER.GT.2)GOTO 444
CALL SETBC(KD,2,3,W)
CALL PATCH(KD,DIFF,W)
CALL PRA(KD,X1,X2,DIFF,W,AP)
DO 190 K=1,KD
DO 130 J=2,NZ
DO 120 I=2,NX
120 W(I,J,K)=3.*W(I,J,K)-DXT12*AP(I,J,K)+DT*A(I,J,K)
130 CONTINUE
DO 180 J=1,NZP
DO 170 I=1,NXP
170 Y(I,J,K)=DXT2*U(I,J,1,K)-AP(I,J,K)+DXD*A(I,J,K)
180 CONTINUE
190 CONTINUE
GOTO 555
C      -- otherwise with Gear method
444 DO 290 K=1,KD
DO 220 J=1,NZP
DO 210 I=1,NXP
210 Y(I,J,K)=0.
220 CONTINUE
DO 280 KO=1,IORDER
CX=DXT*ALF(KO,IORDER)
DO 270 J=1,NZP
DO 260 I=1,NXP

```

```

260 Y(I,J,K)=Y(I,J,K)+CX*U(I,J,K0,K)
270 CONTINUE
280 CONTINUE
290 CONTINUE
CALL SETBC(KD,2,3,Y)
CALL PATCH(KD,DIFF,Y)
CALL PRA(KD,X1,X2,DIFF,Y,AP)
DO 390 K=1,KD
DO 320 J=2,NZ
DO 310 I=2,NX
310 W(I,J,K)=DXT12*Y(I,J,K)-DXTX*AP(I,J,K)+DTB*A(I,J,K)
320 CONTINUE
DO 380 J=1,NZP
DO 370 I=1,NXP
370 Y(I,J,K)=Y(I,J,K)+DXX*A(I,J,K)
380 CONTINUE
390 CONTINUE
555 DO 420 I=1,NXP
XX1=XX(I)
W(I,1,1)=H(1,TT,XX1,BOZH)
W(I,NZP,1)=H(1,TT,XX1,-BOZH)
W(I,1,4)=H(4,TT,XX1,BOZH)
420 W(I,NZP,4)=H(4,TT,XX1,-BOZH)
DO 430 I=1,NZP
ZZ1=ZZ(I)
W(NXP,I,1)=H(1,TT,-BOXH,ZZ1)
430 W(I,1,4)=H(4,TT,BOXH,ZZ1)
CALL CG(KD,AMTX,D,X1,X2,R,P,AP,DIFF
1,LMAX,ERR,L1,ERR1,Y,W)
CALL SETBC(KD,2,3,W)
CALL PATCH(KD,DIFF,W)
KL=KL+L1
WRITE(6,820)KL,L1,ERR1,IORDER
820 FORMAT(/,*ITER*,2(I8),*ERR1*,E14.6,* ORDER *,I6,/)

IF(ERR1.GT.ERR)GOTO 666
DO 600 K=1,KD
DO 540 IK=1,IORD1
IK1=IORDP1-IK
IK2=IK1-1
DO 530 J=1,NZP
DO 520 I=1,NXP
520 U(I,J,IK1,K)=U(I,J,IK2,K)
530 CONTINUE
540 CONTINUE
DO 570 J=1,NZP
DO 560 I=1,NXP
560 U(I,J,1,K)=W(I,J,K)
570 CONTINUE
600 CONTINUE
IF(IORDER.EQ.IXORDER)GOTO 999
DO 650 KKJ=1,IORDER
IF(L.EQ.KST(KKJ))GOTO 888
650 CONTINUE
GOTO 999
888 KKX=IXORD-KKJ
DO 690 K=1,KD
DO 680 J=1,NZP
DO 670 I=1,NXP
670 U(I,J,KKX,K)=W(I,J,K)
680 CONTINUE
690 CONTINUE
IF(L.EQ.KST(IORDER))GOTO 222
GOTO 999
222 L=L/KRATIO

```

```

IORX=IXORD-IORDP1
DO 780 K=1,KD
DO 750 IJK=2,IORDP1
IKK=IORX+IJK
DO 730 J=1,NZP
DO 720 I=1,NXP
720 U(I,J,IJK,K)=U(I,J,IKK,K)
730 CONTINUE
750 CONTINUE
780 CONTINUE
DT=RATIO*DT
GOTO 1111
666 TT=TT-DT
CALL SOL(KD,TT,AP,W)
STOP
END
SUBROUTINE PRA(KD,X1,X2,DIFF,Y,X)
COMMON /FT/NPTS,NSKIP,MTRAN,MSKIP,ISIGN,LOG,IEX
COMMON /EXS/NMAX,EX(4096)
COMMON /SIZE/BOXX,BOXZ,BOXH,BOZH,PIX,PIZ
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
COMMON /PARAM1/PI,DX,DZ,RT2,DT,DXT,DXX
REAL X1(NXPP,1),X2(NXPP,1),WORK(8192)
REAL X(NXPP,NZP,KD),Y(NXPP,NZP,KD),DIFF(NXPP,NZP,KD)
DO 800 K=1,KD
DO 110 J=1,NZP
DO 100 I=1,NXP
100 X2(I,J)=Y(I,J,K)
110 X1(I,J)=Y(I,J,K)
CONTINUE
CALL XDERIV(X1,WORK)
CALL ZDERIV(X2,WORK)
DO 200 J=1,NZP
DO 190 I=1,NXP
190 X2(I,J)=X2(I,J)*DIFF(I,J,K)
200 X1(I,J)=X1(I,J)*DIFF(I,J,K)
CONTINUE
CALL XDERIV(X1,WORK)
CALL ZDERIV(X2,WORK)
DO 270 J=2,NZ
DO 260 I=2,NX
260 X(I,J,K)=DXT*Y(I,J,K)-DXX*(X2(I,J)+X1(I,J))
270 CONTINUE
800 CONTINUE
RETURN
END
SUBROUTINE SETGS(BET,ALF)
REAL BET(6),ALF(6,6)
C --sets Gear coefficients ( BET(2) is coefficient for trapezoidal
   scheme ), FFT parameters and patching arrays
COMMON /PARAM1/PI,DX,DZ,RT2,DT,DXT,DXX
COMMON /SIZE/BOXX,BOXZ,BOXH,BOZH,PIX,PIZ
COMMON /EXS/NMAX,EX(4096)
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
COMMON /FT/NPTS,NSKIP,MTRAN,MSKIP,ISIGN,LOG,IEX
COMMON /GRID/XX(65),ZZ(65)
COMMON /DELS/DDL(65),DDR(65),DELL,DELR
PI=4.*ATAN(1.)
BET(2)=0.5
BET(3)=6./11.
BET(4)=12./25.
BET(5)=60./137.
BET(6)=60./147.
C - first index is for summation and second for order

```

```

ALF(1,3)=18./11.
ALF(1,4)=48./25.
ALF(1,5)=300./137.
ALF(1,6)=360./147.
ALF(2,3)=-9./11.
ALF(2,4)=-36./25.
ALF(2,5)=-300./137.
ALF(2,6)=-450./147.
ALF(3,3)=2./11.
ALF(3,4)=16./25.
ALF(3,5)=200./137.
ALF(3,6)=400./147.
ALF(4,4)=-3./25.
ALF(4,5)=-75./137.
ALF(4,6)=-225./147.
ALF(5,5)=12./137.
ALF(5,6)=72./147.
ALF(6,6)=-10./147.

C      -set FFT parameters
MAXNP=NX+1
NXM=NX-1
NXP=NX+1
NZP=NZ+1
NZM=NZ-1
NXPP=NX+2
NN=NXP*NZP
NTOT=NXPP*NZP
NPTS=2*NX
NMAX=NPTS
NSKIP=1
MTRAN=NZP
MSKIP=NXPP/2
LOG=LOG2(NPTS)
IEX=NMAX/NPTS
BOXX=2.
BOXZ=2.
DX=1./FLOAT(NX)
DZ=1./FLOAT(NZ)
DXX=DX*DX
PIX=PI*DX
PIZ=PI*DZ
BOZH=BOXZ/2.
BOXH=BOXX/2.
WRITE(6,20)NX,NZ,DX,DZ
20 FORMAT(/, * NX*, I4, * NZ*, I5, * DX DZ*, 2(E9.2), /)
DO 40 J=1,NZP
40 ZZ(J)=COS(PIZ*FLOAT(J-1))*BOZH
DO 60 I=1,NXP
60 DDL(I)=DEL(BOXX,NXP,I,NXP)
DDR(I)=DEL(BOXX,1,I,NXP)
XX(I)=COS(PIX*FLOAT(I-1))*BOXH
DELL=DEL(BOXX,NXP,NXP,NXP)
DELR=DEL(BOXX,1,1,NXP)
RETURN
END
SUBROUTINE XDERIV(B,WORK)
COMMON /FT/NPTS,NSKIP,MTRAN,MSKIP,ISIGN,LOG,IEX
COMMON /EXS/NMAX,EX(1)
COMMON /CHEBRL/NCHEB,NSKCHEB,MCHEB,MSKCHEB,FZ
COMMON /SIZE/BOXX,BOXZ,BOXH,BOZH,PIX,PIZ
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
REAL B(NXPP,1),WORK(8192)
NPTS=2*NX
NSKIP=1

```

```

MTRAN=NZP
MSKIP=NXPP/2
ISIGN=-1
LOG=LOG2(NPTS)
IEX=NMAX/NPTS
NCHEB=NX
NSKCHEB=1
MCHEB=NZP
MSKCHEB=NXPP
FZ=1.0/(BOXX*2.*FLOAT(NX))
CALL SRSR(B,EX,WORK)
CALL RLCHBDF(B,B)
ISIGN=1
CALL SRSR(B,EX,WORK)
RETURN
END
SUBROUTINE ZDERIV(B,WORK)
COMMON /FT/NPTS,NSKIP,MTRAN,MSKIP,ISIGN,LOG,IEX
COMMON /EXS/NMAX,EX(1)
COMMON /CHEB/NCHEB,NSKCHEB,MCHEB,MSKCHEB,FZ
COMMON /SIZE/BOXX,BOXZ,BOXH,BOZH,PIX,PIZ
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
REAL B(NXPP,1),WORK(8192)
NPTS=2*NZ
NXHP=NXPP/2
NSKIP=NXHP
MTRAN=NSKIP
MSKIP=1
ISIGN=-1
LOG=LOG2(NPTS)
IEX=NMAX/NPTS
NCHEB=NZ
NSKCHEB=NXHP
MCHEB=NXHP
MSKCHEB=1
FZ=1.0/(BOXZ*2.*FLOAT(NZ))
CALL SCSC(B,EX,WORK)
CALL CHEBDIF(B,B)
ISIGN=1
CALL SCSC(B,EX,WORK)
RETURN
END
FUNCTION Q(K,X,Y)
COMMON /PARAM/ALFA,BETA,CC,CF1
X1=X+2.*FLOAT(K-1)
Q=-CC*(12.*Y*Y-4.)*(EXP(-BETA*X1)+CF1)
RETURN
END
FUNCTION H(K,T,X,Y)
COMMON /PARAM/ALFA,BETA,CC,CF1
X1=X+2.*FLOAT(K-1)
H=EXP(ALFA*T+BETA*X1)+CC*((Y*Y-1.)**2)
RETURN
END
FUNCTION G(K,X,Y)
COMMON /PARAM/ALFA,BETA,CC,CF1
X1=X+2.*FLOAT(K-1)
G=EXP(-BETA*X1)+CF1
RETURN
END
FUNCTION FAP(Z)
COMMON /PARAM/PI,DX,DZ,RT2,DT,DXT,DXX
FAP=1.0/(PI*SQRT(1.-Z*Z))
RETURN

```

```

END
SUBROUTINE CG(KD,AMTX,D,X1,X2,R,P,AP,DIFF
1,LMAX,ERR,L,ERR1,Y,X)
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
REAL X(NXPP,NZP,KD),Y(NXPP,NZP,KD),D(NXPP,NZP,KD)
1,DIFF(NXPP,NZP,KD),AMTX(NXPP,NZP,5,KD)
1,AP(NXPP,NZP,KD),R(NXPP,NZP,KD),P(NXPP,NZP,KD)
2,X1(NXPP,1),X2(NXPP,1)
CALL SETBC(KD,2,3,X)
CALL PATCH(KD,DIFF,X)
CALL PRA(KD,X1,X2,DIFF,X,AP)
DO 30 K=1,KD
DO 20 J=2,NZ
DO 10 I=2,NX
R(I,J,K)=Y(I,J,K)-AP(I,J,K)
CONTINUE
CONTINUE
CALL INVA(KD,AMTX,D,R,X1,P)
RR=0.
DO 50 K=1,KD
DO 40 J=2,NZ
RR=RR+SDOT(NXM,R(2,J,K),1,P(2,J,K),1)
CONTINUE
CONTINUE
L=0
999 L=L+1
CALL SETBC(KD,2,3,P)
CALL PATCH(KD,DIFF,P)
CALL PRA(KD,X1,X2,DIFF,P,AP)
PMP=0.
DO 120 K=1,KD
DO 110 J=2,NZ
PMP=PMP+SDOT(NXM,P(2,J,K),1,AP(2,J,K),1)
CONTINUE
CONTINUE
PMP=RR/PMP
R1R=RR
DO 150 K=1,KD
DO 140 J=2,NZ
CALL SAXPY(NXM,PMP,P(2,J,K),1,X(2,J,K),1)
CALL SAXPY(NXM,-PMP,AP(2,J,K),1,R(2,J,K),1)
CONTINUE
CONTINUE
CALL INVA(KD,AMTX,D,R,X1,AP)
RR=0.
DO 170 K=1,KD
DO 160 J=2,NZ
RR=RR+SDOT(NXM,R(2,J,K),1,AP(2,J,K),1)
CONTINUE
CONTINUE
ERR1=SQRT(RR)
IF(L.GE.LMAX)RETURN
IF(ERR1.LE.ERR)RETURN
PMP=RR/R1R
DO 230 K=1,KD
DO 220 J=2,NZ
DO 210 I=2,NX
P(I,J,K)=AP(I,J,K)+PMP*P(I,J,K)
CONTINUE
CONTINUE
GOTO 999
END
SUBROUTINE INVA(KD,AMTX,D,Y,R,X)
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP

```

```

REAL AMTX(NXPP,NZP,5,KD),D(NXPP,NZP,KD)
REAL Y(NXPP,NZP,KD),R(NXPP,1),X(NXPP,NZP,KD)
DO 800 K=1,KD
R(1,1)=Y(2,2,K)/D(1,1,K)
DO 20 I=2,NXM
R(I,1)=(Y(I+1,2,K)-AMTX(I,1,2,K)*R(I-1,1))/D(I,1,K)
DO 80 J=2,NZM
R(1,J)=(Y(2,J+1,K)-AMTX(1,J,1,K)*R(1,J-1))/D(1,J,K)
DO 70 I=2,NXM
R(I,J)=(Y(I+1,J+1,K)-AMTX(I,J,2,K)*R(I-1,J))
1-AMTX(I,J,1,K)*R(I,J-1))/D(I,J,K)
CONTINUE
X(NX,NZ,K)=R(NXM,NZM)
DO 160 I=1,NXM-1
X(NX-I,NZ,K)=R(NXM-I,NZM)-AMTX(NXM-I,NZM,4,K)*X(NXP-I,NZ,K)
2/D(NXM-I,NZM,K)
DO 180 J=1,NZM-1
X(NX,NZ-J,K)=R(NXM,NZM-J)-AMTX(NXM,NZM-J,5,K)*X(NX,NZP-J,K)
3/D(NXM,NZM-J,K)
DO 170 I=1,NXM-1
X(NX-I,NZ-J,K)=R(NXM-I,NZM-J)-(AMTX(NXM-I,NZM-J,4,K)
1*X(NXP-I,NZ-J,K)+AMTX(NXM-I,NZM-J,5,K)*X(NX-I,NZP-J,K))
2/D(NXM-I,NZM-J,K)
CONTINUE
CONTINUE
RETURN
END
SUBROUTINE CR(KD,S,AMTX,D)
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
COMMON /PARAM1/PI,DX,DZ,RT2,DT,DXT,DXX
REAL AMTX(NXPP,NZP,5,KD),D(NXPP,NZP,KD)
DO 800 K=1,KD
D(1,1,K)=S*(AMTX(1,1,3,K)+DXT)
DO 10 I=2,NXM
D(I,1,K)=S*(AMTX(I,1,3,K)+DXT)
1-AMTX(I-1,1,4,K)*AMTX(I,1,2,K)/D(I-1,1,K)
DO 40 J=2,NZM
D(I,J,K)=S*(AMTX(I,J,3,K)+DXT)
2-AMTX(I,J-1,5,K)*AMTX(I,J,1,K)/D(I,J-1,K)
DO 20 I=2,NXM
D(I,J,K)=S*(AMTX(I,J,3,K)+DXT)
3-AMTX(I-1,J,4,K)*AMTX(I,J,2,K)/D(I-1,J,K)
4-AMTX(I,J-1,5,K)*AMTX(I,J,1,K)/D(I,J-1,K)
CONTINUE
CONTINUE
RETURN
END
SUBROUTINE SETMAT(KD,AMTX)
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
COMMON /SIZE/BOXX,BOXZ,BOXH,BOZH,PIX,PIZ
COMMON /GRID/XX(65),ZZ(65)
COMMON /PARAM1/PI,DX,DZ,RT2,DT,DXT,DXX
REAL AMTX(NXPP,NZP,5,KD)
DO 800 K=1,KD
DO 50 J=1,NZM
YF=COS(PIZ*(FLOAT(J)+.5))*BOZH
YB=COS(PIZ*(FLOAT(J)-.5))*BOZH
YFB=ZZ(J+1)
Y1=FAP(YB)
Y2=.5*FAP(YFB)
Y3=FAP(YF)
Y21=Y2*Y1
Y23=Y3*Y2
DO 40 I=1,NXM

```

```

XF=COS(PIX*(FLOAT(I)+.5))*BOXH
XB=COS(PIX*(FLOAT(I)-.5))*BOXH
XFB=XX(I+1)
X1=FAP(XB)
X2=.5*FAP(XFB)
X3=FAP(XF)
X21=X2*X1
X23=X2*X3
AMTX(I,J,5,K)=-(G(K,XF,YF)+G(K,XB,YB))*Y23
AMTX(I,J,1,K)=-(G(K,XF,YB)+G(K,XB,YB))*Y21
AMTX(I,J,4,K)=-(G(K,XF,YF)+G(K,XF,YB))*X23
AMTX(I,J,2,K)=-(G(K,XB,YF)+G(K,XB,YB))*X21
40 50 CONTINUE
DO 90 J=1,NZM
DO 80 I=1,NXM
80 AMTX(I,J,3,K)=-AMTX(I,J,5,K)-AMTX(I,J,1,K)
1-AMTX(I,J,4,K)-AMTX(I,J,2,K)
90 CONTINUE
DO 130 J=1,NZM
AMTX(NXM,J,4,K)=0.
130 AMTX(1,J,2,K)=0.
DO 150 J=1,NXM
AMTX(J,1,1,K)=0.
150 AMTX(J,NZM,5,K)=0.
800 CONTINUE
RETURN
END
SUBROUTINE DCS(KD,DG,DIFF,A)
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
COMMON /SIZE/BOXX,BOXZ,BOXH,BOZH,PIX,PIZ
COMMON /GRID/XX(65),ZZ(65)
REAL DIFF(NXPP,NZP,KD),A(NXPP,NZP,KD),DG(1)
NN=NXM*NZM
DO 800 K=1,KD
DO 50 J=1,NZP
ZZ1=ZZ(J)
DO 40 I=1,NXP
XX1=XX(I)
A(I,J,K)=Q(K,XX1,ZZ1)
40 DIFF(I,J,K)=G(K,XX1,ZZ1)
50 CONTINUE
DO 70 J=1,NZP
J1=(J-1)*NXP
DO 60 I=1,NXP
60 DG(I+J1)=DIFF(I,J,K)
70 CONTINUE
KMX=ISMAX(NN,DG,1)
KMN=ISMN(NN,DG,1)
DIFMAX=DG(KMX)
DIFMIN=DG(KMN)
IF(DIFMIN.LE.0.0)GOTO 111
RRD=DIFMAX/DIFMIN
111 WRITE(6,170)K,DIFMAX,DIFMIN,RRD
170 FORMAT(/,* SUBDOMAIN*,15,* DIFMAX DIFMIN *,
12(E14.6),* RATIO *,E14.6,/)
800 CONTINUE
RETURN
END
SUBROUTINE SOL(KD,TT,AP,W)
COMMON /PARAM1/PI,DX,DZ,RT2,DT,DXT,DXX
COMMON /SIZE/BOXX,BOXZ,BOXH,BOZH,PIX,PIZ
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
COMMON /GRID/XX(65),ZZ(65)
REAL AP(NXPP,NZP,KD),W(NXPP,NZP,KD)

```

```

DO 800 K=1,KD
DO 30 J=1,NZP
ZZ1=ZZ(J)
DO 10 I=1,NXP
XX1=XX(I)
AP(I,J,K)=H(K,TT,XX1,ZZ1)
10 CONTINUE
ER=0.
DO 780 J=2,NZ
DO 770 I=2,NX
770 ER=ER+ABS(AP(I,J,K)-W(I,J,K))
780 CONTINUE
ER=ER*DX*DZ
WRITE(6,880)TT,K,ER
880 FORMAT(/,* TIME *,E16.6,* SUBDOMAIN *,14,* ERROR*,E16.6,/)

800 CONTINUE
RETURN
END
SUBROUTINE PATCH(KD,DIFF,X)
COMMON /DELS/DDL(65),DDR(65),DELL,DELR
COMMON /SIZE/BOXX,BOXZ,BOXH,BOZH,PIX,PIZ
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
REAL X(NXPP,NZP,KD),DIFF(NXPP,NZP,KD),DL(65,4),DR(65,4)
DO 170 K=1,KD-1
DO 50 J=1,NZP
DL(J,K)=0.
DR(J,K)=0.
X(1,J,K)=0.
X(NXP,J,K+1)=0.
DO 40 I=1,NXP
C - right derivative
DR(J,K)=DR(J,K)+DDR(I)*X(I,J,K)
C - left derivative
DL(J,K)=DL(J,K)+DDL(I)*X(I,J,K+1)
40 CONTINUE
50 CONTINUE
DO 120 J=1,NZP
X(1,J,K)=(DIFF(1,J,K)*DR(J,K)-DIFF(NXP,J,K+1)*DL(J,K))
1/(DIFF(NXP,J,K+1)*DELL-DIFF(1,J,K)*DELR)
X(NXP,J,K+1)=X(1,J,K)
120 CONTINUE
170 CONTINUE
RETURN
END
SUBROUTINE SETBC(KD,K1,K2,X)
C ---- sets Neumann boundary condition dT/dy = 0 at upper and lower
C boundary in subdomains K1 and K2
COMMON /DELS/DDL(65),DDR(65),DELL,DELR
COMMON /SIZE/BOXX,BOXZ,BOXH,BOZH,PIX,PIZ
COMMON /N/NXPP,NXP,NZP,NX,NZ,NXM,NZM,NTOT,MAXNP
REAL X(NXPP,NZP,KD),DL(65,4),DR(65,4)
DO 50 I=1,NXP
DR(I,K1)=0.
DR(I,K2)=0.
DO 40 J=2,NZP
C - right derivative { upper }
DR(I,K1)=DR(I,K1)+DDR(J)*X(I,J,K1)
DR(I,K2)=DR(I,K2)+DDR(J)*X(I,J,K2)
40 CONTINUE
50 CONTINUE
DO 120 I=1,NXP
X(I,1,K1)=-DR(I,K1)/DELR
X(I,1,K2)=-DR(I,K2)/DELR
120 CONTINUE

```

```

DO 150 I=1,NXP
DL(I,K1)=0.
DL(I,K2)=0.
DO 140 J=1,NZ
DL(I,K1)=DL(I,K1)+DDL(J)*X(I,J,K1)
DL(I,K2)=DL(I,K2)+DDL(J)*X(I,J,K2)
CONTINUE
140
150
CONTINUE
DO 160 I=1,NXP
X(I,NZP,K1)=-DL(I,K1)/DELL
X(I,NZP,K2)=-DL(I,K2)/DELL
160
CONTINUE
DO 57 I=1,NXP
DR(I,K1)=0.
DR(I,K2)=0.
DO 47 J=2,NZP
C      - right derivative { upper }
DR(I,K1)=DR(I,K1)+DDR(J)*X(I,J,K1)
DR(I,K2)=DR(I,K2)+DDR(J)*X(I,J,K2)
47
57
CONTINUE
CONTINUE
DO 127 I=1,NXP
X(I,1,K1)=-DR(I,K1)/DELR
X(I,1,K2)=-DR(I,K2)/DELR
127
CONTINUE
RETURN
END
FUNCTION DEL(XL,K,J,N)
KR=K-1
JR = J - 1
NR = N - 1
FAC = 1.
IF( KR .NE. NR ) GO TO 10
KR = 0
JR = NR - JR
FAC = -1.
10
DEL = FAC*XNUM(KR,JR,NR)/DEN(KR,JR,NR)
DEL = DEL*(2./XL)
RETURN
END
FUNCTION DEN(KR,JR,NR)
COMMON /PARAM1/PI,DX,DZ,RT2,DT,DXT,DXX
IF(KR .EQ. 0) GO TO 20
DEN = FLOAT(NR)*SIN(PI*FLOAT(KR)/FLOAT(NR))
IF(JR .EQ. 0 .OR. JR .EQ. NR) DEN = DEN*2.
20
RETURN
DEN = .5*FLOAT(NR)
IF(MOD(JR,2) .EQ. 1) DEN = - DEN
IF(JR .EQ. 0 .OR. JR .EQ. NR) DEN = 1.
RETURN
END
FUNCTION XNUM(KR,JR,NR)
COMMON /PARAM1/PI,DX,DZ,RT2,DT,DXT,DXX
IF(KR .EQ. 0) GO TO 20
XNUM = FF(KR+JR,NR) + FF(KR-JR,NR)
20
RETURN
IF(JR .EQ. 0 .OR. JR .EQ. NR) GO TO 30
XNUM = .5*(FLOAT(NR)+.5)*(  

1 *(FLOAT(NR)+.5) + (COT(PI*FLOAT(JR)/FLOAT(2*NR)))**2)  

2 + 1./8. - .25/(SIN(PI*FLOAT(JR)/FLOAT(2*NR)))**2)  

3 - 5*FLOAT(NR*NR)
30
RETURN
XNUM = .5
IF( MOD(NR,2) .EQ. 1 ) XNUM = - XNUM

```

```

IF(JR.EQ.0) XNUM = (1./3.)*FLOAT(NR*NR) + 1./6.
RETURN
END
FUNCTION FF(I,NR)
COMMON /PARAM1/PI,DX,DZ,RT2,DT,DXT,DXX
IF(I .EQ. 0) GO TO 20
FF = FLOAT(NR)*.5/TAN(PI*FLOAT(I)/FLOAT(2*NR))
IF(MOD(I,2) .EQ. 0) FF = -FF
20
RETURN
FF = 0.
RETURN
END

```

ALFA $0.10E+01$ BETA- $0.10E+01$ CC $0.10E+01$ CFL $0.10E+01$

TIME 0.1 DTMAX 0.01 RATIO $10.$ S 1.1 LMAX 100 ERR $1.E-05$

NX 8 NZ 8

SUBDOMAIN	1	DIFMAX	DIFMIN	$0.371E+01$	$0.136E+01$	RATIO	$0.271E+01$
SUBDOMAIN	2	DIFMAX	DIFMIN	$0.210E+02$	$0.371E+01$	RATIO	$0.567E+01$
SUBDOMAIN	3	DIFMAX	DIFMIN	$0.149E+03$	$0.215E+02$	RATIO	$0.708E+01$
SUBDOMAIN	4	DIFMAX	DIFMIN	$0.109E+04$	$0.149E+03$	RATIO	$0.734E+01$

Cumulative number of iterations 229

Number of iterations in the last step 11

Error in the last step (global) $0.784695E-05$

Highest order of time integration 4

TIME	0.1	SUBDOMAIN	1	ERROR	$0.904854E-05$
TIME	0.1	SUBDOMAIN	2	ERROR	$0.167686E-04$
TIME	0.1	SUBDOMAIN	3	ERROR	$0.529883E-05$
TIME	0.1	SUBDOMAIN	4	ERROR	$0.123279E-06$

need explanation
OK

REFERENCES

- Naumann, R. J. 1982 An Analytical Approach to Thermal Modeling of Bridgman-Type Crystal Growth. Parts I, II. *J. Crystal Growth* 58, 554-584.
- Moro, B. & Orszag, S. A. 1984 Spectral Methods for Multi-Domain Diffusion Problems. To be published.
- Orszag, S. A. 1980 Spectral Methods for Problems in Complex Geometries. *J. Comp. Phys.* 36, 70-92.
- Shih, A. & Orszag, S. A. 1984 Order and Disorder in Two-Dimensional Double Diffusive Convection. To be published.

SUPPLEMENT

to

NASA CONTRACT NAS8-35838

January 28, 1985

Prepared by:

Fred C. Hart Associates, Inc.
1110 Vermont Avenue, N.W.
Washington, D.C. 20005

In Figs 1-6, we plot the salt and temperature contours for a flow with

$$\alpha_1 = 1$$

$$\alpha_2 = -200$$

$$\beta_1 = -1$$

$$\beta_2 = -1$$

$$\kappa_1 = 0.12172$$

$$\kappa_2 = 1.2172$$

$$v = 1.2172$$

Here the results plotted in Figs 1-2 are obtained on a 16x16 spectral grid at $t = 0.6$ after evolution from the initial conditions $T=S=0$, $u = \sin x \cos y$, $v = -\cos x \sin y$. The results plotted in Figs 3-4 are for a 32x32 grid while those in Figs 5-6 are for a 64x64 grid. Comparison of these figures shows the excellent convergence properties of the spectral code.

SALT CONTOURS 16 x 16 spectral code

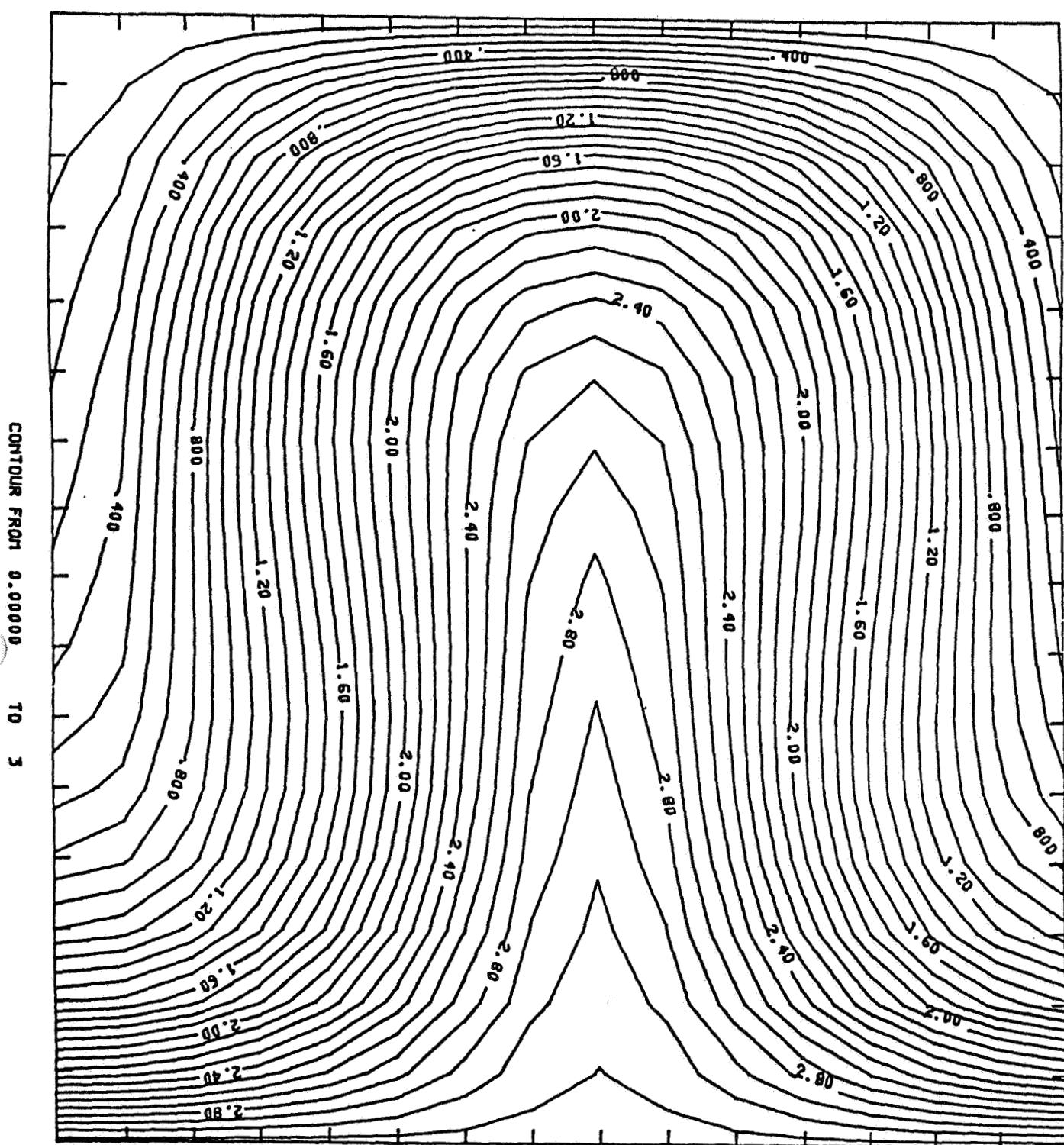


Figure 1

TEMPERATURE CONTOURS

16 x 16 Spectral Code

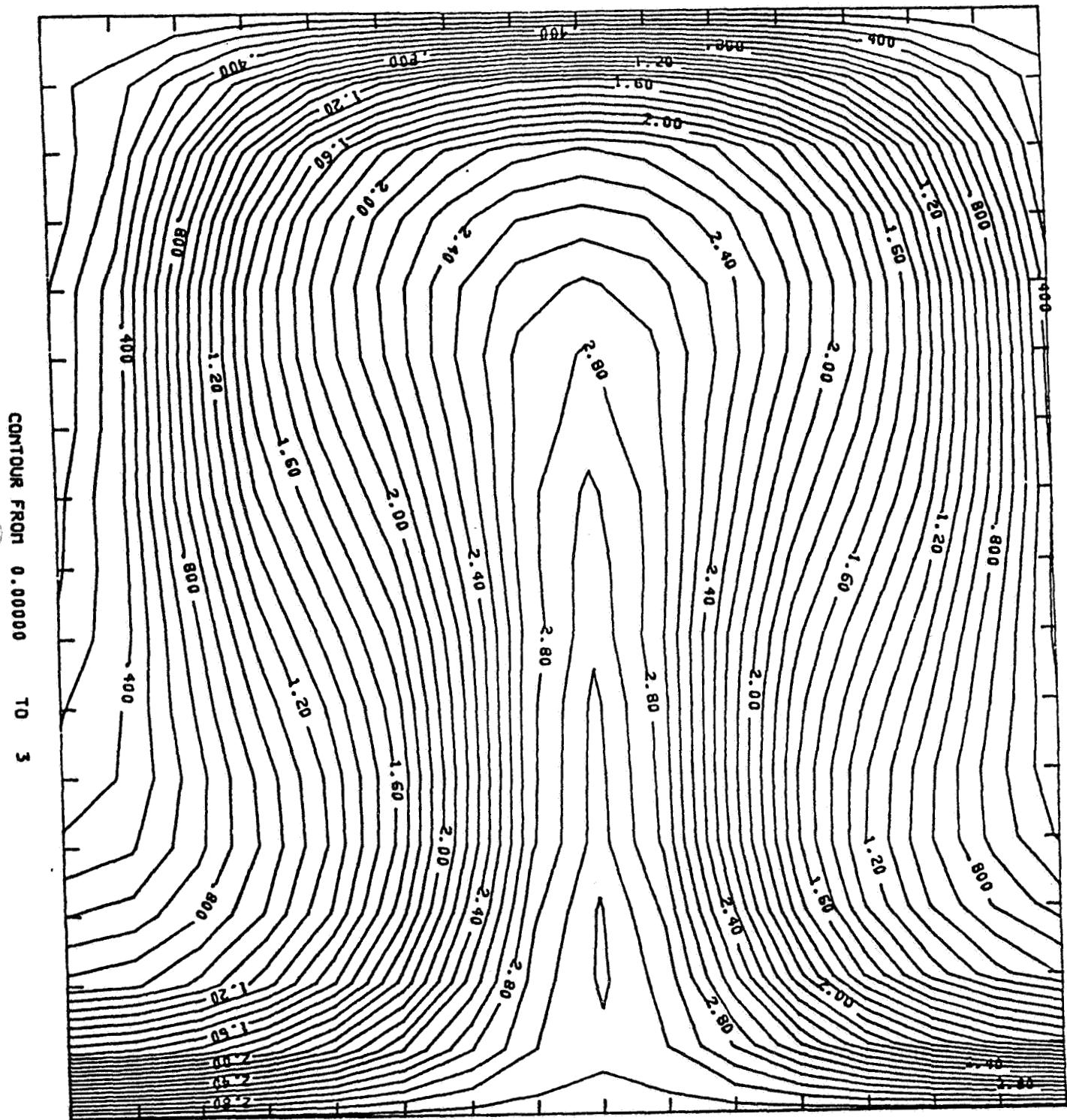
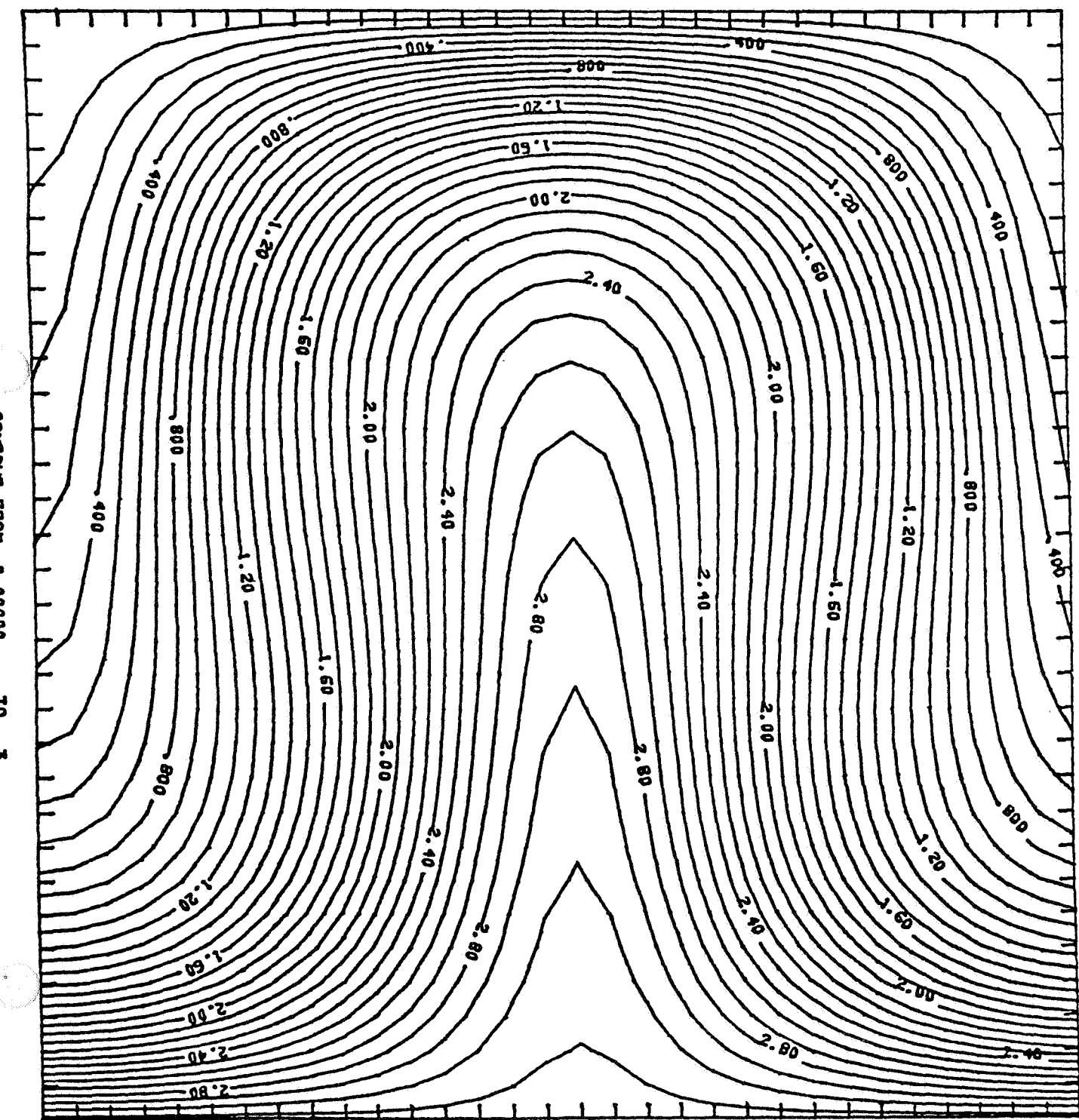


Figure 2

Figure 3

SALT CONTOURS

32 x 32 Spectral Code



TEMPERATURE CONTOURS

32 x 32 Spectral Code

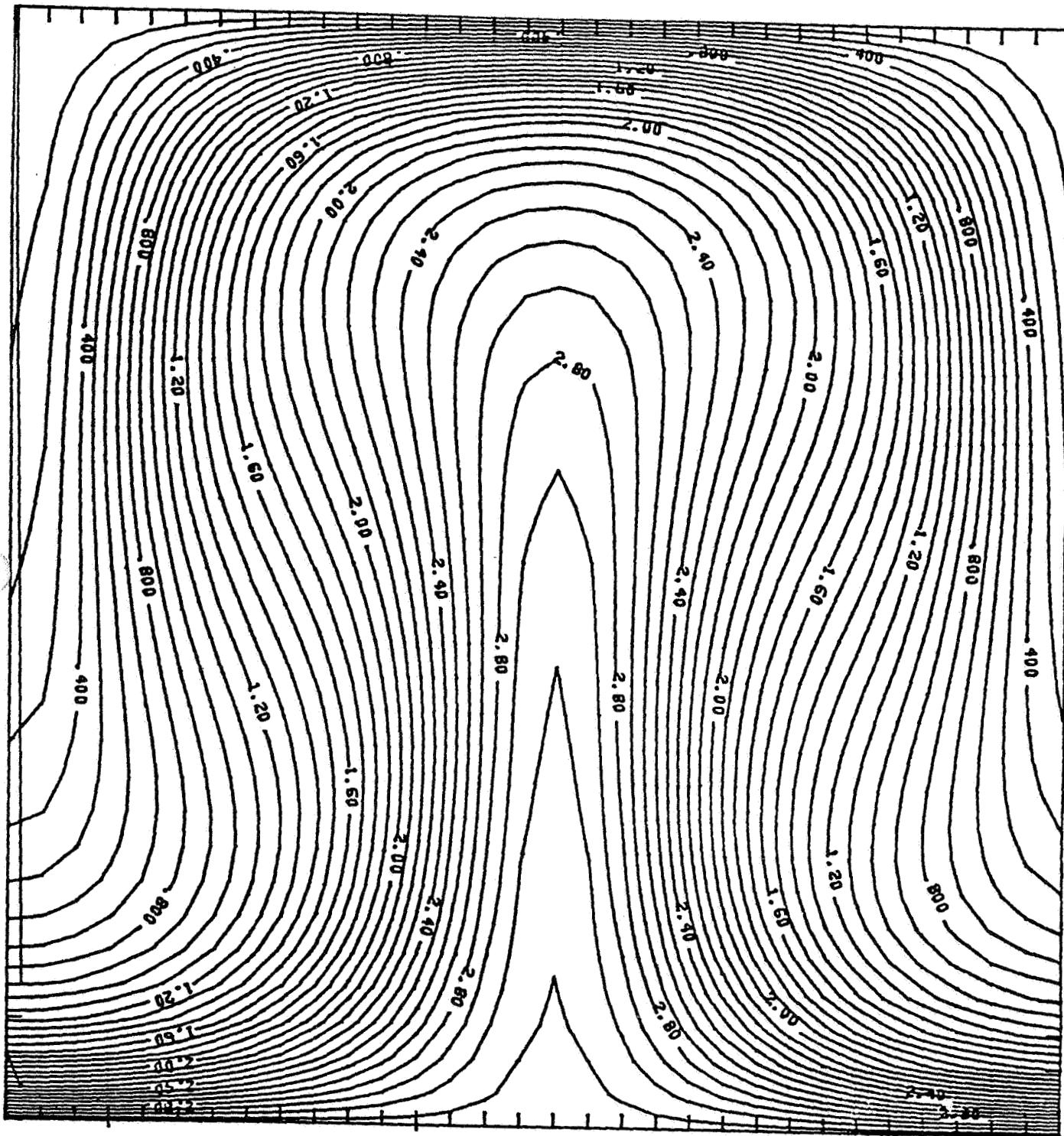


Figure 4

SALT CONTOURS

64 x 64 Spectral Code

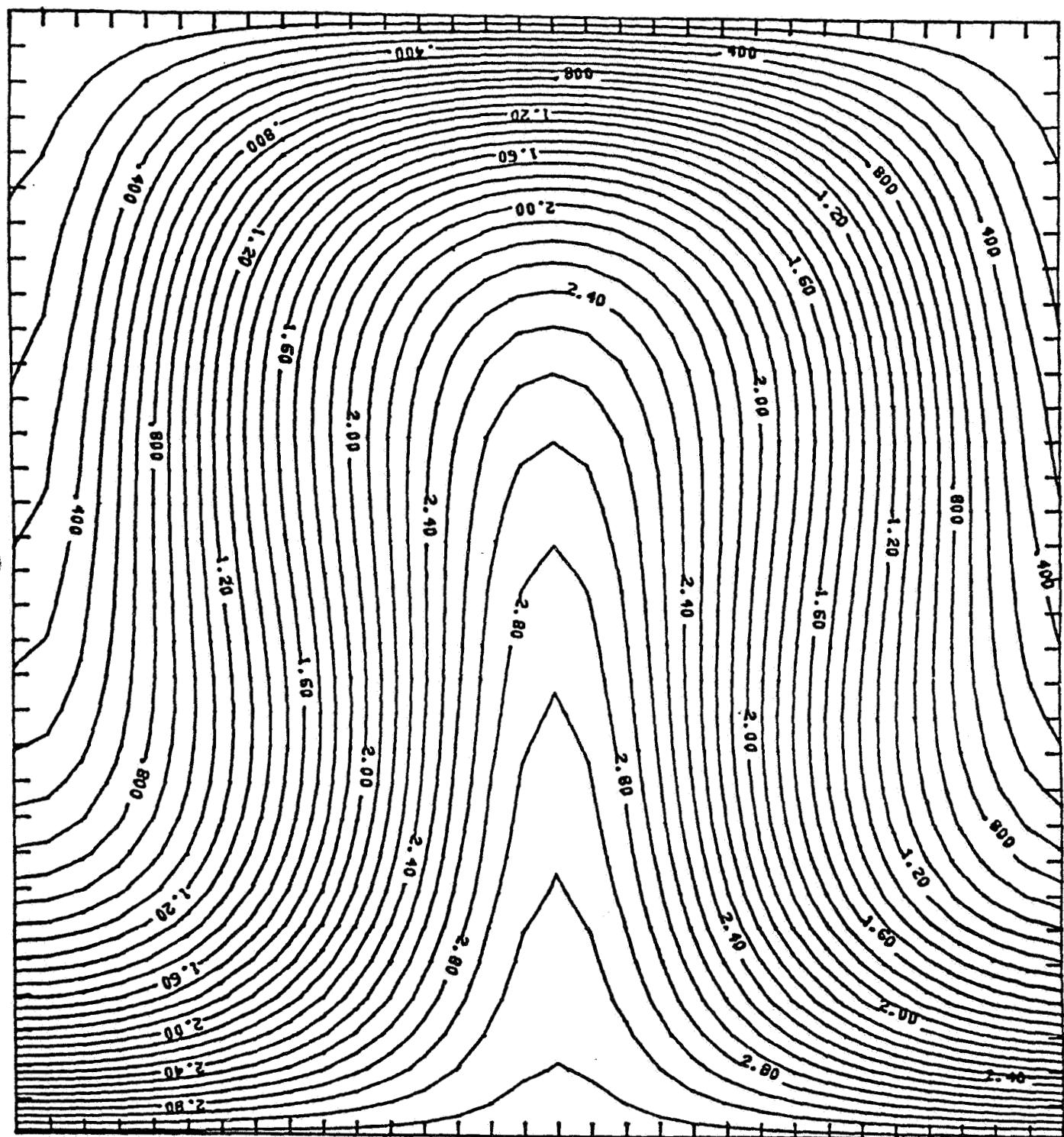


Figure 5

TEMPERATURE CONTOURS

64 x 64 Spectral Code

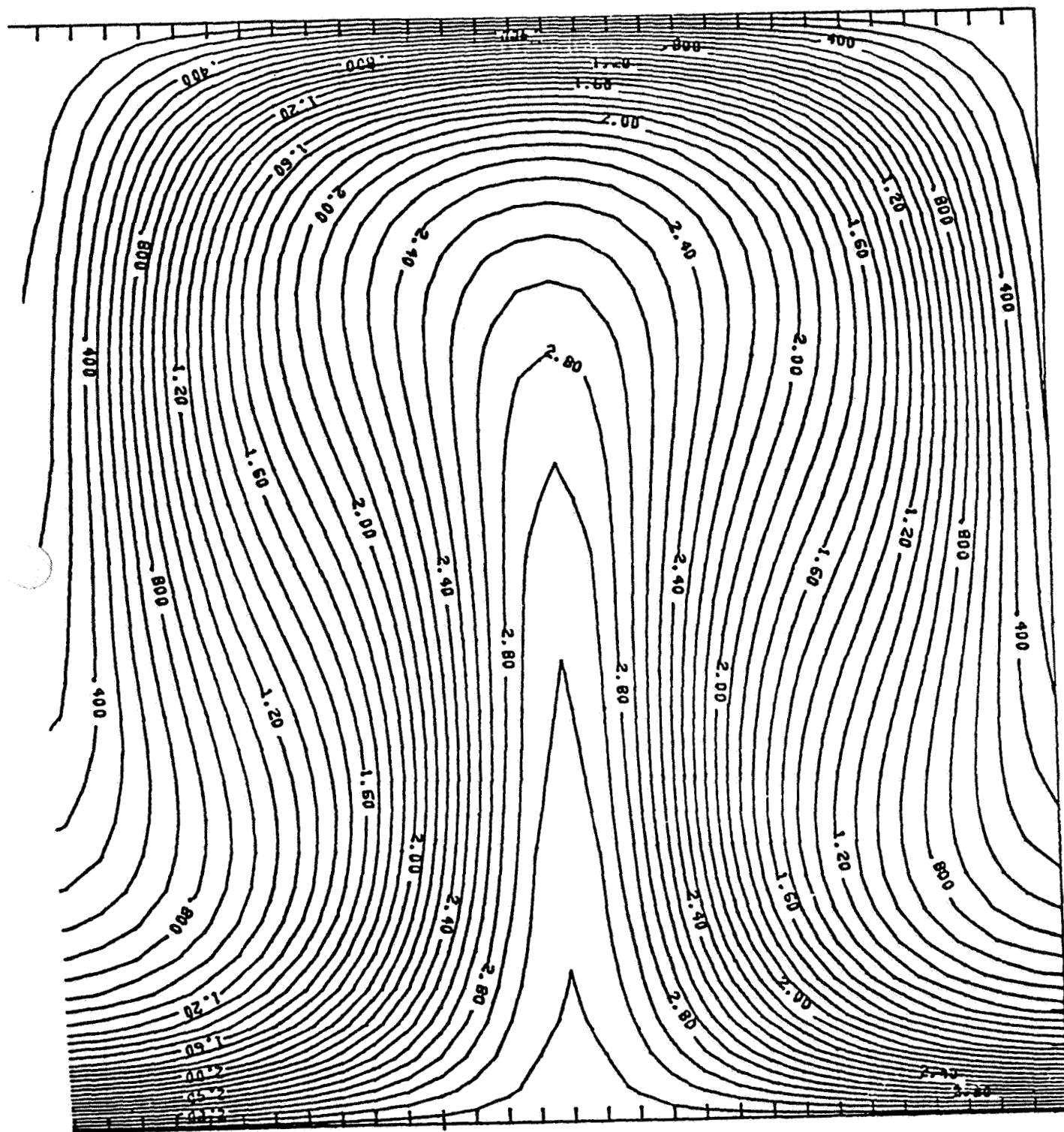


Figure 6

The problem that is solved is

$$\frac{\partial H}{\partial t} = \frac{\partial}{\partial x} G \frac{\partial H}{\partial x} + Q$$

$$Q = -c(12y^2 - 4) e^{-\beta x} + \frac{\alpha}{\beta^2}$$

$$G = e^{-\beta x} + \frac{\alpha}{\beta^2}$$

$$H = e^{\alpha t + \beta x} + c(y^2 - 1)^2$$

in four patched x- domains. The results are listed below for an 8x8 spectral grid. Here DIFMAX = Max G, DIFMIN = Min G, and ERROR is the maximum difference between the computed and exact solutions.

ALFA 0.10E+01 BETA=0.10E+01 CC 0.10E+01 CFL 0.10E+01											
TIME	0.1	DIMAX	0.01	RATIO	10.	S	1.1	LMAX	100	ERR	1.E-05
NX	8	NZ	8								
SUBDOMAIN	1	DIFMAX	DIFMIN	0.371E+01	0.136E+01	RATIO	0.271E+01				
SUBDOMAIN	2	DIFMAX	DIFMIN	0.210E+02	0.371E+01	RATIO	0.567E+01				
SUBDOMAIN	3	DIFMAX	DIFMIN	0.149E+03	0.215E+02	RATIO	0.708E+01				
SUBDOMAIN	4	DIFMAX	DIFMIN	0.109E+04	0.149E+03	RATIO	0.734E+01				
Cumulative number of iterations 229											
Number of iterations in the last step 11											
Error in the last step (global) 0.784695E-05											
Highest order of time integration 4											
TIME	0.1	SUBDOMAIN	1	ERROR	0.904854E-05						
TIME	0.1	SUBDOMAIN	2	ERROR	0.167686E-04						
TIME	0.1	SUBDOMAIN	3	ERROR	0.529883E-05						
TIME	0.1	SUBDOMAIN	4	ERROR	0.123279E-06						